

DTIC FILE COPY

AD-A202 571



DESIGN AND DEVELOPMENT OF A  
COMPUTER-BASED MESSAGE TRANSFER SYSTEM  
FOR THE AIR FORCE LOGISTICS COMMAND  
PACKET RADIO NETWORK

THESIS  
William J. Tavis, Captain, USAF  
AFIT/GE/ENG/88D-53

DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY

**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

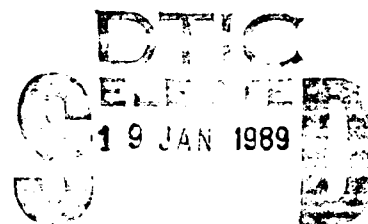
This document has been approved  
for public release and sale in  
distribution is unlimited.

DTIC  
ELECTE  
19 JAN 1989  
S D  
E

AFIT/GE/ENG/88D-53

DESIGN AND DEVELOPMENT OF A  
COMPUTER-BASED MESSAGE TRANSFER SYSTEM  
FOR THE AIR FORCE LOGISTICS COMMAND  
PACKET RADIO NETWORK

THESIS  
William J. Taris, Captain, USAF  
AFIT/GE/ENG/88D-53



Approved for public release; distribution unlimited

AFIT/GE/ENG/88D-53

DESIGN AND DEVELOPMENT OF A  
COMPUTER-BASED MESSAGE TRANSFER SYSTEM FOR THE  
AIR FORCE LOGISTICS COMMAND PACKET RADIO NETWORK

THESIS

Presented to the Faculty of the School of Engineering  
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science in Electrical Engineering

William J. Taris, B.S.E.E.

Captain, USAF

December 1988

Approved for public release; distribution unlimited

## Acknowledgements

This thesis would not have been possible without the help provided by my faculty advisor, LTC Albert B. Garcia. His patience and guidance enabled me to gain a working knowledge of engineering principles. I am also indebted to Capt Thomas Saner, who contributed his enthusiasm for this topic, his time and, working with the Air Force Logistics Command, helped in obtaining hardware resources. Finally, I wish to thank my spouse, Dianne, for her continued help and understanding.

William J. Taris

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/ _____	
Availability Codes	
Dist	Avail and/or Special
A-1	



## Table of Contents

	<u>Page</u>
Acknowledgements . . . . .	ii
List of Figures . . . . .	vi
List of Tables . . . . .	viii
Abstract . . . . .	ix
 I. Introduction . . . . .	1.1
Background . . . . .	1.1
Problem Statement . . . . .	1.6
Scope and Limitations . . . . .	1.7
Approach and Presentation . . . . .	1.9
 II. Requirements . . . . .	2.1
Introduction . . . . .	2.1
System Reference . . . . .	2.1
Software Project Constraints . . . . .	2.3
Requirements Listing . . . . .	2.4
Message Processing Requirements . . . . .	2.4
Operator Interface Requirements . . . . .	2.5
Customer Interface Requirements . . . . .	2.6
Logical System Model . . . . .	2.8
System Overview DFD . . . . .	2.8
Generate Message DFD . . . . .	2.10
Transmit Message DFD . . . . .	2.11
Receive Message DFD . . . . .	2.13
Validation Criteria . . . . .	2.16
Comparison Between Requirements Listing and Logical System Model . . . . .	2.16
Validation Tests . . . . .	2.19
Expected Software Response . . . . .	2.19
Conclusion . . . . .	2.22
 III. System Design . . . . .	3.1
Introduction . . . . .	3.1
System Objectives . . . . .	3.1
Major Design Constraints . . . . .	3.2
Design Description . . . . .	3.3
Data Description . . . . .	3.4
Message Data Structure . . . . .	3.4
Routing Table Data Structure . . . . .	3.7
Program Structure . . . . .	3.8
System Structure Chart . . . . .	3.9
Changing System States . . . . .	3.12

## Table of Contents (Cont'd)

	<u>Page</u>
Modules . . . . .	3.14
State Transition Manager Module . . . . .	3.14
Initialize System State Module . . . . .	3.15
Initialize Port Module . . . . .	3.16
Initialize TNC Module . . . . .	3.17
Initialize Linked Lists Module . . . . .	3.18
Operator Interface State . . . . .	3.19
Display Menu and Prompt Module . . . . .	3.21
Prepare a Message for Transmit Module . . . . .	3.23
Access Archive Module . . . . .	3.25
Generate Message . . . . .	3.30
Receive State Module . . . . .	3.31
Receive Message Module . . . . .	3.31
Update Routing Table Module (Receive) . . . . .	3.33
Update Received Message Archive Module . . . . .	3.35
Transmit State Module . . . . .	3.35
Transmit Message Module . . . . .	3.37
Update Routing Table Module (Transmit) . . . . .	3.37
Update Transmitted Message Archive Module . . . . .	3.39
Reset System State Module . . . . .	3.40
Reset TNC Module . . . . .	3.41
Close Port Module . . . . .	3.41
Store Linked Lists Module . . . . .	3.42
Error Recovery and Exception Handling . . . . .	3.42
Error Recovery . . . . .	3.42
Exception Handling . . . . .	3.44
Design Verification . . . . .	3.46
IV. Testing . . . . .	4.1
Introduction . . . . .	4.1
Verification Testing . . . . .	4.2
Unit Testing . . . . .	4.2
Integration Testing . . . . .	4.7
Validation Testing . . . . .	4.10
Summary . . . . .	4.11
V. Results, Conclusions, and Recommendations . . . . .	5.1
Introduction . . . . .	5.1
Test Results . . . . .	5.1
Verification Tests . . . . .	5.1
Validation Tests . . . . .	5.2
Conclusions . . . . .	5.2
Recommendations for Further Study . . . . .	5.3

Table of Contents (Cont'd)

	<u>Page</u>
Bibliography . . . . .	BIB.1
Appendix A: Advanced Electronic Applications Model PK-232 Terminal Node Controller .	A.1
Appendix B: User's Manual . . . . .	B.1
Appendix C: System Structure Charts . . . . .	C.1
Appendix D: Data Dictionary . . . . .	D.1
Appendix E: Test Plan . . . . .	E.1
Appendix F: Test Results . . . . .	F.1
Appendix G: Computer Program Code . . . . .	G.1
Vita . . . . .	V.1

## List of Figures

Figure	Page
1. Map of US Showing Location of AFLC PRN's Eight Nodes . . . . .	1.4
2. Equipment Configuration at Each of the Eight AFLC PRN Nodes . . . . .	1.5
3. System Overview Data Flow Diagram . . . . .	2.7
4. Generate Message Data Flow Diagram . . . . .	2.10
5. Transmit Message Data Flow Diagram . . . . .	2.12
6. Receive Message Data Flow Diagram . . . . .	2.14
7. Representation of the Message Data Structure Broken Down by Fields . . . . .	3.5
8. Representation of the Routing Table Data Structure Broken Down by Fields . . . . .	3.8
9. System Structure Chart . . . . .	3.10
10. Flowchart Depicting the System State Change Process . . . . .	3.13
11. Initialize System State Structure Chart . . . . .	3.16
12. Operator Interface State Structure Chart . . . . .	3.22
13. Receive State Structure Chart . . . . .	3.32
14. Transmit State Structure Chart . . . . .	3.36
15. Reset System State Structure Chart . . . . .	3.40
C-1 System Structure Chart . . . . .	C.1
C-2 Initialize System State Module Structure Chart . . . . .	C.2
C-3 Initialize TNC Module Structure Chart . . . . .	C.2
C-4 Initialize Linked Lists Module Structure Chart . . . . .	C.3
C-5 Operator Interface State Module Structure Chart . . . . .	C.3

List of Figures (Cont'd)

Figure	Page
C-6 Display Menu and Prompt Module Structure Chart . . . . .	C.4
C-7 Prepare Message for Transmit Module Structure Chart . . . . .	C.4
C-8 Access Archive Module Structure Chart . . . . .	C.5
C-9 Generate Message Module Structure Chart . . . . .	C.6
C-10 Receive State Module Structure Chart . . . . .	C.6
C-11 Receive Message Module Structure Chart . . . . .	C.7
C-12 Update Routing Table (Receive) Module Structure Chart . . . . .	C.7
C-13 Update Received Message Archive Module Structure Chart . . . . .	C.8
C-14 Transmit State Module Structure Chart . . . . .	C.8
C-15 Transfer Message Module Structure Chart . . . . .	C.9
C-16 Update Routing Table (Transmit) Module Structure Chart . . . . .	C.9
C-17 Update Transmitted Message Archive Module Structure Chart . . . . .	C.10
C-18 Reset System State Module Structure Chart . . . . .	C.10

### List of Tables

Table		Page
I.	Comparison Between the Requirements Listing and the Logical System Model . . . . .	2.17
II.	Requirement, Validation Test, and Expected Software Response . . . . .	2.19
III.	Design Verification Table . . . . .	3.47
F-I	Alpha Phase Validation Tests, Expected Software Responses, Actual Responses . . . . .	F.11

Abstract

The Air Force Logistics Command <sup>(AFLC)</sup> Packet Radio Network (PRN) is a specialized communications network that enables communication between eight logistics command centers throughout the continental United States. The PRN communicates by transferring a message from a microcomputer onto a broadcast radio channel. This thesis effort designs, and partially develops the design for, a computer-based message transfer system that operates the PRN. First, system requirements are established, a logical system model is constructed, and validation tests are detailed. The computer-based message transfer system has two fundamental requirements--that it be easy to use and that it provide automatic routing through the network. Next, the design is built supporting a hierarchical program structure, modularity and information hiding. After the computer code, detailed by design, is written, it is tested. Testing involves a comparison of the validation tests detailed earlier with the computer program's operation. The results show that this thesis effort resulted in an operational computer-based message system for the PRN that satisfies the two fundamental requirements of ease of use and automatic routing. (17) (—

DESIGN AND DEVELOPMENT OF A COMPUTER-BASED  
MESSAGE TRANSFER SYSTEM FOR THE  
AIR FORCE LOGISTICS COMMAND PACKET RADIO NETWORK

I. Introduction

Background

Military communication systems are recognized force multipliers [19:1]. This recognition leads to continued growth in the number of military communication systems. As additional military communication systems are implemented, significant contention for a limited resource, bandwidth, demands efficient utilization of that resource. Paralleling the growth in military communication systems is the dramatic increase in the use of military computers. This use led almost immediately to the need for a capability to communicate between various military computers. These computer communications are data transfers, typically of a burst-type nature, some requiring use of a radio channel. A pair of computers communicating in a burst-type nature over a dedicated radio channel is an inefficient use of the bandwidth resource because most of the time the channel is idle.

In response to the need for a computer communications capability over a radio channel, packet radio is used [10:24-31]. Packet radio allows a group of computers to communicate over a shared radio channel. Packet radio uses the radio channel as a multiple-access channel.

The radio channel is a multiple-access channel because many nodes will want to send message traffic over the same channel, possibly at the same time. One efficient multiple-channel access technique to use for this type of communication is Carrier Sense Multiple Access (CSMA), a communication method based on the transmission of packets between computer nodes [11:1400-1402;12:9].

A packet is a maximum length number of data characters along with a fixed number of control characters. The length of a packet is a maximum of 256 bytes of data, 8 bits per byte, and control characters [7]. The PRN has an average message length of 1500 characters, 1500 bytes, so that one message is broken up into a large number of packets. Consequently, the channel transmission time used by an individual packet is much shorter than the transmission time for transmitting the entire message all at once. Packetized transmission of a message allows sharing of the radio channel. The radio channel can be shared at the end of every packet transmission, unlike the situation of transmitting the entire message where sharing of the channel is possible only at the end of a message's transmission. The sharing of the radio channel between packet transmissions is accomplished by interleaving in time the packets from all nodes on the radio channel. Another important advantage of packet transmission is that corruption of the data during

transmission causes the retransmission of the corrupted packet, whereas the corruption of data during transmission of the entire message requires that the entire message be retransmitted.

A computer node consists of the computer and associated radio equipment. The computer node is simply referred to as a node. With CSMA, when a node has a packet to transmit it senses the radio channel. If the channel is not busy, the node transmits the packet. If the node senses that the channel is busy, it reschedules the transmission for a later time when it again starts the channel-sensing process. The process of transmitting packets over the radio channel continues until the entire message is transmitted. Communication systems that use this type of CSMA technique are called packet radio networks (PRNs). AFLC's PRN is such a network.

AFLC's PRN is a specialized communications network that provides message communication between eight Logistics Command Centers dispersed throughout the continental United States, see Figure 1. The network is used for contingency operations and is included in the AFLC's Survival, Recovery, and Reconstitution Plan [3].

AFLC's PRN replaced a system that used radio voice links. The previous system was slow and personnel-intensive because the network's messages were manually encrypted for voice transmission. The PRN is faster and requires fewer

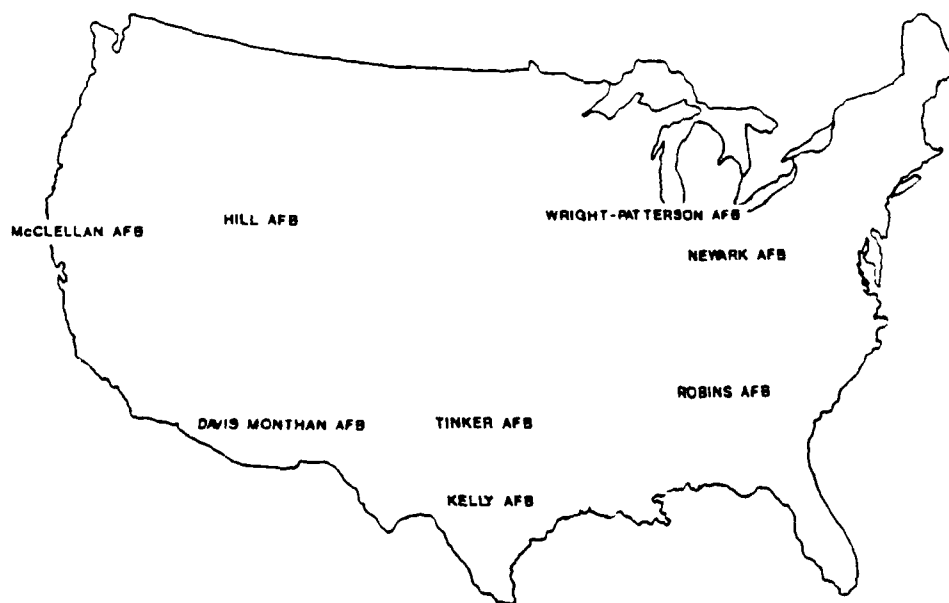


Figure 1. Map of U.S. Showing Location of AFLC PRN's Eight Nodes

personnel than the previous system. An increase in speed is possible because encryption is performed automatically as the message leaves the computer. Unattended operation is possible with the PRN.

Each PRN node consists of an IBM PC-compatible computer, a computer program, an encryption device, a terminal node controller, and a single-frequency, high frequency (HF) radio, see Figure 2. The computer program is responsible for interfacing with node operators requiring the transmission or receipt of PRN messages. The computer

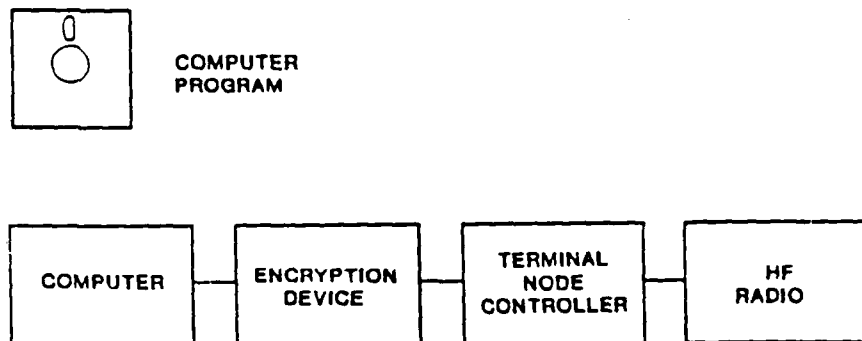


Figure 2. Equipment Configuration at Each of the Eight AFLC PRN Nodes

program is also responsible to interface with the terminal node controller so that messages are transmitted over, and received from, the radio channel. The encryption device is viewed as a transparent device that rearranges the data stream between the computer and terminal node controller. A terminal node controller is similar in operation to a modem with the added requirements of operating with a radio, using packet radio techniques, instead of a telephone line (see Appendix A).

When a message is transmitted between a source and a destination node, the node operator instructs the computer

program to enter and transmit the message. The receiving node, under computer control, accepts the message into storage and indicates to that node operator that a message has arrived. This scenario views the network as being fully connected, each node able to establish a direct link with every other node. Because the PRN uses an HF radio channel, full connectivity cannot be assumed at all transmission times. In a situation of less than full connectivity, a method must be instituted to use intermediate nodes to forward the message to the destination. This method is referred to as a network routing algorithm [22:539-541].

Currently, the PRN has only a basic computer program available. The current network routing algorithm is a manual system requiring complete operator intervention. The PRN needs a computer-based message transfer system with an associated automated network routing algorithm.

#### Problem Statement

AFLC's PRN lacks a computer-based message transfer system (MTS) with an associated network routing algorithm. The goal of this thesis effort is to design and develop an MTS for the AFLC's PRN.

### Scope and Limitations

The goal of this thesis effort is to provide an MTS. The MTS will provide the capability to transmit a message from any node to any other node over the shared HF radio channel. Central to providing this capability is the need for a network routing algorithm.

The MTS will have the ability to generate messages, transmit messages, and receive messages. These three abilities form the baseline for the MTS.

The MTS will satisfy three overall objectives:

- (1) Portability - the MTS will be able to operate on any IBM PC-compatible computer running MS-DOS 3.X or higher.
- (2) Flexibility - the MTS will have an ability for growth, should its requirements expand.
- (3) Maintainability - all system documentation will be complete and concise. Changes made to the MTS should involve no hardware modifications.

The network routing algorithm will be implemented in software resident at each computer node. The choices for implementation of the network routing algorithm are either a software or a firmware implementation. Choosing the software implementation satisfies the three overall objectives for the MTS:

- (1) Portability - the software implementation will be independent of the type of IBM PC-compatible computer being used.

- (2) Flexibility - the ease with which a software implementation can be modified aids in its flexibility.
- (3) Maintainability - because a fix to a problem code segment can be remedied by inserting a different floppy disk with no hardware changes involved aids maintainability.

A firmware implementation would involve a firmware product residing within the TNC. The TNC model chosen for the PRN does not have commercially available firmware as do some TNCs [14]. The firmware for this project would have to be totally developed in-house.

Interoperability of the MTS with other networks will not be developed during this thesis effort. However, it is realized that the PRN, being a U.S. military communication network, requires interoperability. The objective of flexibility has, as one of its benefits, the ability to enhance the PRN to interoperate.

This thesis effort will not review the type of logistics information that will be incorporated into a message's text. The message text will simply be viewed as a series of ASCII characters.

The encryption of messages will not be investigated nor developed into the MTS. The PRN provides for encryption of the message traffic by a computer expansion card that interfaces between the data stream input and output of the computer and the data stream input and output of the TNC.

For the purpose of this thesis effort, the encryption device will be assumed to be completely transparent. Because the expansion device is viewed as completely transparent, there is no need for incorporation into the MTS development. The MTS will be expected to perform only on the PRN not having the encryption devices installed.

#### Approach and Presentation

The initial approach is to perform a literature search for information about related PRN system designs. Specifically, this review will help to identify problem areas that other investigators have encountered. Publications and periodicals available in the AFIT library are the starting point for this investigation.

The goal of this project is to provide an MTS. The MTS project involves three areas of development: detail specific requirements of the message transfer system identifying from those specific requirements that the MTS will satisfy; design of the MTS computer program; and testing of the MTS computer program.

Specific requirements of the message transfer system are documented as a plain English description. Following this, the logical system model highlights the data flow of the MTS. The logical system model is represented by data-flow diagrams and a data dictionary. Validation criteria that

check the ability of the completed MTS to satisfy the system requirements are developed and presented.

Design of the MTS develops the logical system model data-flow diagrams into structure charts through the use of transform analysis. A bottom-up programming style turns the structure charts into computer code.

The Test Plan referenced in the chapter on testing specifies the steps taken to verify the MTS computer program. Validation testing for the MTS computer program uses the validation criteria developed during requirements analysis.

## II. Requirements

### Introduction

Satisfying the PRN's need for a message transfer system computer program with an associated network routing algorithm (MTS) begins by an investigation of specific PRN requirements. A baseline for the start of this effort is the knowledge that the PRN's message-handling processes need to generate messages, transmit messages, and receive messages.

After a brief discussion of the PRN system reference and software project constraints, a detailed listing of the specific requirements for the packet radio network is presented in plain English. Following this, a logical system model, using data-flow diagrams and a data dictionary, indicate the data flow through the MTS. The final section in this chapter establishes the validation criteria that will be used to compare the PRN requirements against the completed MTS.

System Reference. The purpose of AFLC's PRN is to provide secure record communications during a lack of contingency communications capabilities needed to support natural disasters and the wartime Survival, Recovery, Reconstitution scenario.

Each of the eight Logistics Command Centers, Wright-Patterson, Hill, Robins, Kelly, McClellan, Tinker, Davis-Monthan Air Force Bases, and Newark Air Force Base, Ohio, is

a PRN node, see Figure 1. Each node will be equipped with a Tempest Z-150 microcomputer, an encryption device, a terminal node controller, and an HF AN/URC-119 radio, see Figure 2. Each node will use a copy of the MTS computer program, developed as part of this thesis, on the Z-150 microcomputer to operate on the PRN.

Generation of the messages for transmission is accomplished on an off-line IBM PC-compatible computer and stored on a 5 1/4-inch disk. Once the message is generated, it is delivered to the PRN operator. The PRN operator processes the message by transferring it from the 5 1/4-inch disk to the transmit section of the MTS. At the receiving node, the message is automatically received and the destination node operator is informed that a message has arrived.

Due to the characteristics of HF radio transmission, continuous direct radio contact between all pairs of PRN nodes is not expected. There is a need for the PRN to transmit messages from any node to any other node through the use of intermediate nodes acting as repeaters. The PRN's ability to automatically make use of intermediate nodes as repeaters is the goal of the network routing algorithm. This ability enables the transmission of a message that is unable to go directly from source node to destination node by automatically choosing a transmission path using an intermediate node or nodes to aid in the transmission of the message.

Software Project Constraints. The MTS software development is constrained by three factors: the volume of message traffic that can be processed, the computer hardware equipment configuration, and the limited development time.

The volume of message traffic that can be processed by each node is limited by the radio transmission rate of 300 baud. This transmission rate results in a channel capacity of 1,080,000 bits/hr where the network is modeled as two nodes, one transmitting and one receiving. Knowing the packet sizes used by AX.25 and that an average message is 1500 characters results in a requirement of 17070 bits/msg. The error-free channel throughput is calculated by dividing the channel capacity by the average message size. The error-free channel throughput is calculated to be 63 msgs/hr. Errors occurring in 2% of the packets cause the throughput to be 58 msgs/hr.

The Z-150 computer is configured with two floppy disk drives and 512k bytes of random access memory (RAM). These storage limitations cause some software implementation choices to be disregarded.

Time limits on the development of the MTS dictate that priority requirements are identified for full-scale incorporation into a completed MTS. This points out that not all the PRN requirements are to be incorporated into the MTS computer program developed as a part of this thesis effort.

### Requirements Listing

Requirements for the MTS computer program were developed after discussions with the PRN system administrator and a system operator. Information was exchanged regarding the way the older manual system worked and about the expectations for an automated system. The requirements specified serve to establish a complete set of requirements for the PRN.

The requirements listing is broken into three areas. The three areas are message processing requirements, operator interface requirements, and customer interface requirements. These three areas correspond to the focus of the message transfer systems's operation.

References to "system" in the following requirements listing mean either an individual node operation performed at each node or an overall PRN function.

Message Processing Requirements. These requirements focus on the entire PRN operation.

(1) The system shall transmit a message from one of eight nodes to another node, more than one node, or all nodes. The capability shall exist to transmit messages between all eight nodes.

(2) Messages shall be composed of 10 fields. These fields will be message source, message destination(s), message author, date/time, priority, security classification, message text, time of transmission, and time message was received.

(3) Messages shall be composed of, at most, one and a half pages of text.

(4) A network routing algorithm shall be used by the system to determine the best route to transmit a message. The network routing algorithm shall be automated. The network routing algorithm shall provide a route for transmission of a message even when there is not a direct connection between the source and destination.

(5) Messages shall be encrypted for transmission.

(6) The system shall provide archival storage of messages that are transmitted or received. Following a message transmission and before storing the message, a transmission time will be appended to the message. Following a message being received and before storing the message, a time of receipt will be appended to the message.

(7) The system shall provide a positive acknowledgement that a message has been received at the destination node.

(8) The system shall provide a transmit message queue. The transmit message queue should be capable of holding TXQUEUEMAX messages.

(9) Queued messages not able to be transmitted after TIMEOUT minutes or TRYOUT tries shall be flagged for the operator's attention.

(10) The system shall transmit messages based on priority.

(11) The system shall provide for receiving messages, whether the operator position is attended or unattended.

(12) Complete system documentation shall be provided. This documentation shall include a detailed explanation of the computer program, points of contact, and reference documentation.

Operator Interface Requirements. These requirements focus on the system operator interface.

(13) Use of the operator interface shall not inhibit receiving messages.

(14) The system operator interface shall be menu driven with English instructions and help features for all major operator options.

(15) The system shall provide the operator with a message completeness indication as messages are being entered.

(16) The system operator shall be provided with the ability to enter messages for transmission into the transmission message queue. This ability shall be provided for up to TXQUEUEMAX messages.

(17) The system operator shall be able to access a record of all node activity for the previous 24 hours. This record shall be provided in hard copy form. Node activity is defined as transmitted messages, received messages, and instances where the node was used as an intermediate node. Selectable options shall provide choices among how the records will be formatted for output.

(18) A converse mode shall be provided. Converse mode allows interactive communication between two system operators.

(19) A system operator handbook shall be provided. The handbook shall detail all menu options and provide sample screen displays. The handbook shall reproduce in hard copy form all help feature displays.

Customer Interface Requirements. These requirements focus on the customer's generation of messages.

(20) The system shall provide a method for a customer to generate a message. The method shall be a computer program that displays a message template and queries the customer for input.

(21) The Generate Message Program shall be menu driven with English instructions and help features for all customer options.

(22) The information that shall be provided by the customer when generating a message is the source, destination, author, date/time, priority, security classification, subject, and message text.

(23) The Generate Message Program shall be portable to any IBM PC-compatible computer running MS-DOS 3.X or higher. The Generate Message Program will produce the message onto a 5 1/4-inch disk.



### Logical System Model

The purpose of the logical system model is to describe data flows associated with the MTS [9:8-24]. The logical system model uses data-flow diagrams (DFD) and a data dictionary to describe MTS data flows. The DFDs presented in this section detail the basic DFD logical levels.

System Overview DFD. The purpose of the system overview DFD is to provide a framework for subsequent development. No specific data dictionary entries are associated with this DFD.

The system overview DFD, Figure 3, represents the highest level DFD of the MTS. The three primary processes involved with the MTS--generate messages, transmit messages, receive messages--are each represented.

The generate message process is represented on the DFD as the customer's message data store. This data store signifies the process of generating a message, transferring it to a 5 1/4-inch disk, and delivering the disk to the system operator.

The transmit message process is represented by the top half of the system overview DFD. This basic DFD description shows the three steps involved with transmitting a message. The first step is to transfer the message from the customer's 5 1/4-inch disk into a priority-ordered transmit queue. This step is represented by the DFD process TRANSFER

MESSAGE. The second step sets up the next queued message's path through the PRN. This step is represented by the DFD process SET UP PATH. The final step in the transmit message process is to segment the message. Because the message is larger than one packet, it must be segmented into packets. This step is represented by the DFD process SEGMENT MESSAGE OUT.

The receive message process, the final primary MTS process, is represented by the bottom half of the System Overview DFD. Three steps are involved with receiving a message. The first step is recognition of the received message path that has been established. This step is represented by the DFD process RECOGNIZE PATH. The second step is to receive and store incoming message packets. This step is represented by the DFD process SEGMENT MESSAGE IN. The final step in the receive message process is to notify the system operator that a message has been received. This step is represented by the DFD process NOTIFY OPERATOR.

Common to the transmit and receive message processes is the interfacing between the computer and the TNC. This interfacing is represented by the DFD process INTERFACE TO TERMINAL NODE CONTROLLER.

The three primary processes involved with the MTS have been presented as a basic DFD representation, the system overview DFD. Each of the three primary processes, generate

message, transmit message, and receive message, are now presented as separate DFDs. Unlike the system overview DFD, these DFDs have associated data dictionary entries. The data dictionary is presented in Appendix D. These three separate DFDs, still at a basic logic level, serve to illustrate the primary MTS processes.

Generate Message DFD. The purpose of the generate message process is to provide the customer with a method to generate a message and transfer that message onto a 5 1/4-inch disk. Figure 4 is a DFD of the generate message process. Central to this process is the display of a

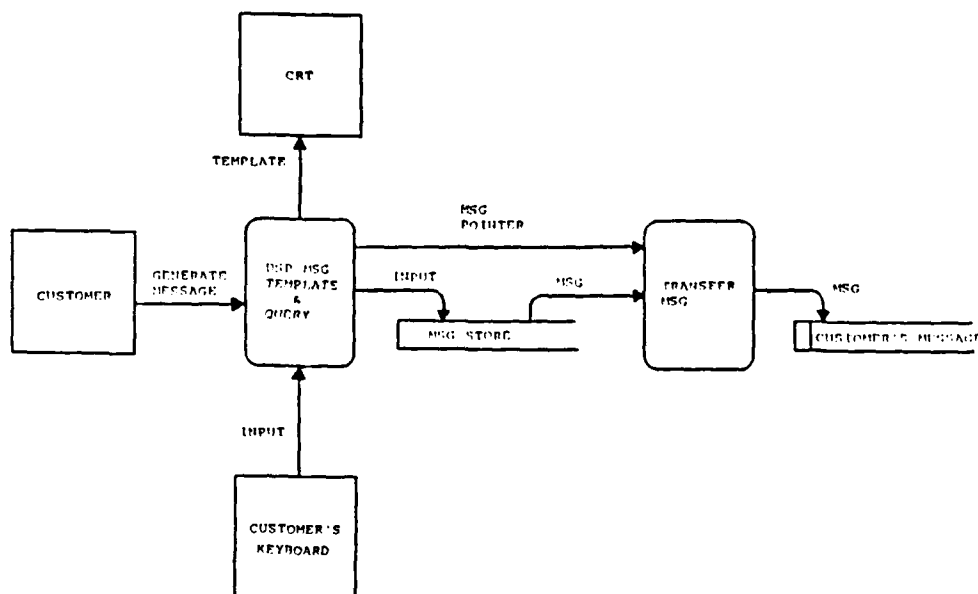


Figure 4. Generate Message Data Flow Diagram

message template onto the customer's computer screen and querying the customer for keyboard inputs. Once all message fields are queried and the customer exits and the entire message is transferred onto a 5 1/4-inch disk.

Transmit Message DFD. The purpose of the transmit message process is to transfer a message from a customer's 5 1/4-inch disk to a distant node. The transmit message process requires an operator. The transmit message DFD is shown as Figure 5. Earlier, the transmit message process was described as a three-step procedure: transfer message, set up message path, and segment the message. Each of these three steps involves two distinct DFD processes.

The transfer step checks that the message being requested by the operator for entry is complete, storing completed messages in the computer's RAM. The message's priority then drives the ordering of memory location pointers to the next message to be transmitted from the transmit message queue. These two transfer message steps are represented by the DFD processes CHECK FOR COMPLETENESS and TQUEUE.

The message path procedure for the transmit message function begins as a check of the transmit message pointers. Finding the next message to be transmitted in the transmit queue results in that message's destination being passed to

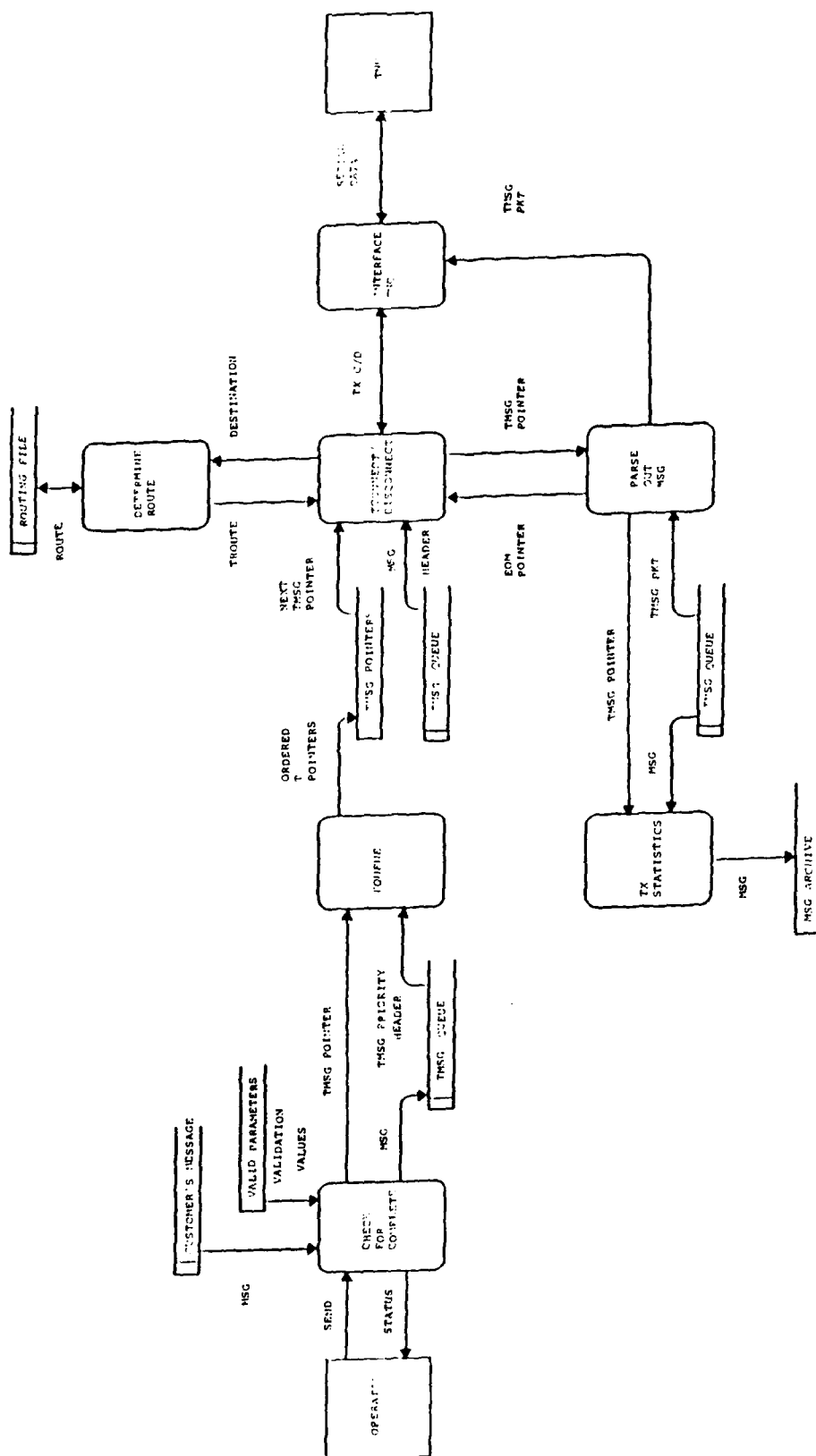


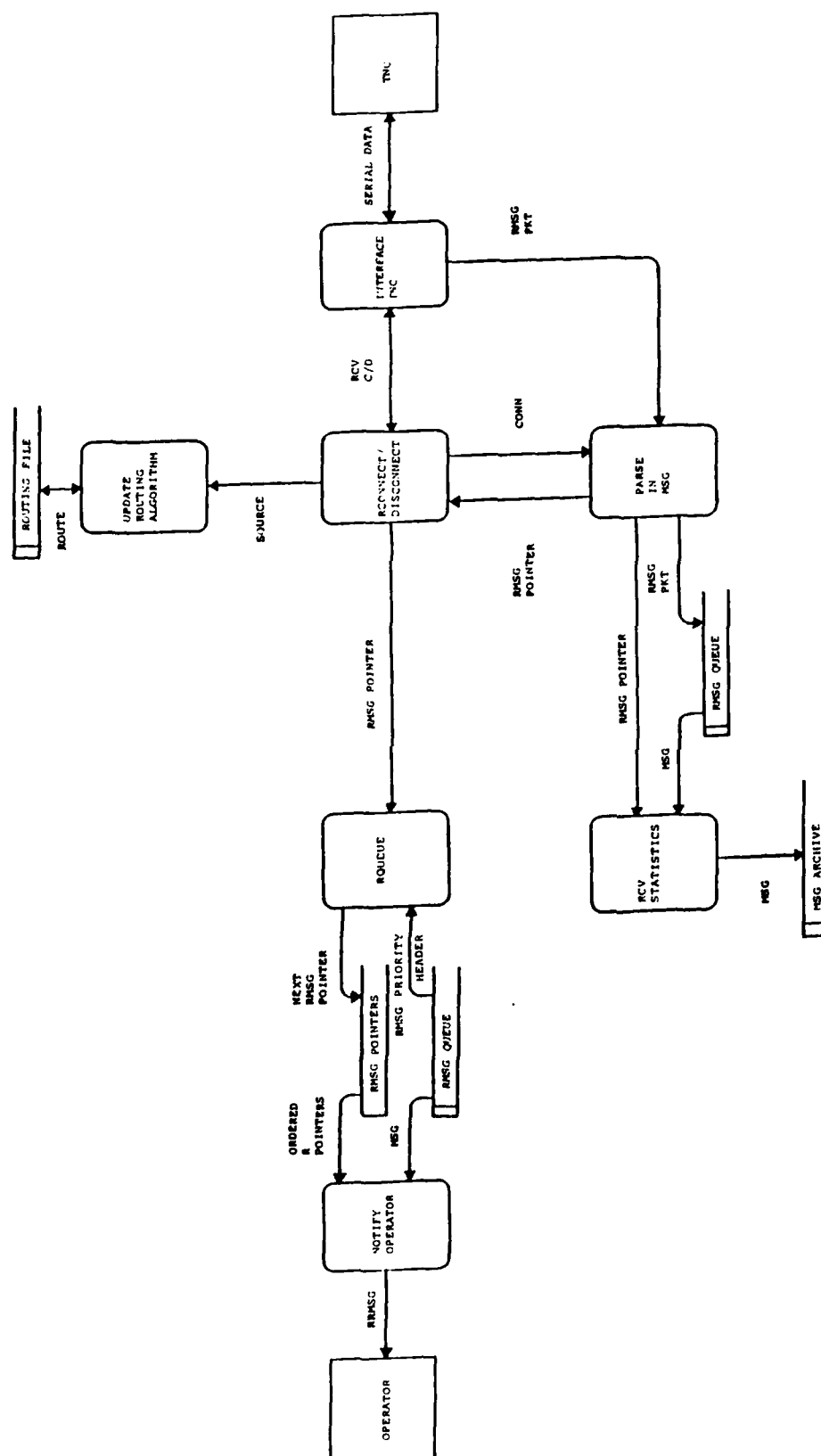
Figure 5. Transmit Message Data Flow Diagram

the network routing algorithm. The network routing algorithm produces a transmission path. This transmission path is then used to establish a connection between source and destination node. These operations are represented as DFD processes TCONNECT/DISCONNECT and DETERMINE ROUTE.

Once a connection is established, the segmenting message procedure begins to segment the message into packets. These packets are then transferred to the TNC for transmission. This is represented by the DFD process PARSE OUT MESSAGE. After the entire message has been transferred, a transmission time is appended to the message and the message is stored on a 5 1/4-inch disk. This process is represented by the DFD process TX STATISTICS.

The interface between the transmit message process and the TNC is represented by the DFD process INTERFACE TNC. The primary purpose of the INTERFACE TNC process is responsibility for control and data transfer from the computer to the terminal node controller.

Receive Message DFD. The purpose of the receive message process is to receive a message from a distant node and inform the operator of a received message. A message will be received regardless of whether or not the operator position is attended. The received message DFD is shown as Figure 6. Earlier, the message process was described as a



three-step procedure: recognize message path, receive message packets, and notify operator. Each of these three steps involves two distinct DFD processes.

The recognize message path step establishes that a received message is arriving. This is represented by the DFD process RCONNECT/DISCONNECT. The receive message packet step accepts message packets arriving, and stores them as partial message segments. This is represented by the DFD process PARSE IN MESSAGE. When all message packets have been received, a receive time is appended to the message. The message is then stored on a 5 1/4-inch disk. These two operations are represented by the DFD process RCV STATISTICS. The newly received message path is used to update the network routing algorithm. This is represented by the DFD process UPDATE ROUTING ALGORITHM.

Once a message is completely received, the receive message queue pointers are reordered, based on priority. This is represented by the DFD process RQUEUE. Next, an indication of a received message is passed to the operator's computer screen if the operator position is attended. This process is represented by the DFD process NOTIFY OPERATOR.

The interface between the receive message process and the TNC is represented by the DFD process INTERFACE TNC. Operations performed under this process are similar to those described for the transmit message process.

### Validation Criteria

Validation criteria answer the question of what message transfer system requirements are satisfied by the completed MTS [13:670-674]. The validation process has two steps. First, a comparison is made between the message transfer system requirements and the logical system model. Next, validation tests are detailed.

### Comparison Between Requirements Listing and Logical System Model.

A comparison of the message transfer system requirements listing with the logical system model is presented in Table I. Requirements that are identified as not being satisfied by the logical system model are valid message transfer system requirements that have been identified but are not able to be satisfied by this thesis effort.

Table I indicates that requirements 5, 7, and 18 will not be satisfied by this thesis effort. The requirement for encryption, requirement 5, is accomplished by an expansion card placed in the Z-150. The expansion card is secure communications equipment. The administrative overhead associated with secure communications equipment placed this requirement beyond the scope of this thesis effort.

The requirement for a positive acknowledgement that a message has been read by the system operator at the destination node, requirement 7, is not satisfied by the MTS. A

manual procedure that requires the operator to positively acknowledge all received messages could be mandated to satisfy this requirement.

The requirement of a converse mode, requirement 18, between system operators is not satisfied by the MTS.

Table I. Comparison Between the Requirements Listing and the Logical System Model.

Requirement	Logical System Model			Not Satisfied	Comments
	TX	RCV	GEN		
(1)	X	X			Transfer to all nodes
(2)			X		Additional Msg Info
(3)			X		1.5 Pages of Text
(4)	X				Routing Algorithm
(5)				X	Encryption
(6)	X	X			Transmit Stats
(7)				X	Positive Ack
(8)	X				Queue
(9)	X	X			TIMEOUT/TRYOUT
(10)	X				Priority Handling
(11)		X			Unattended Receipt

Table I. Comparison Between the Requirements Listing and the Logical System Model (Cont'd).

Requirement	: Logical System Model	: Not		
	: TX	: RCV	: GEN	: Satisfied
				: Comments
(12)	: X	: X	: X	: : Sys
				: : Documentation
(13)	: X	: X		: : Noninhibiting
				: : Operator
				: : Interface
(14)	: X	: X		: : Operator
				: : Position
				: : Menu-Driven
(15)	: X			: : Complete-
				: : ness
				: : Check
(16)	: X	: X		: : Queueing
(17)	: X	: X		: : Archive
				: : of Msg
				: : Traffic
(18)				: : X Converse
				: : Mode
(19)	: X	: X		: : Operator
				: : Handbook
(20)			: X	: : Customer
				: : Generate
				: : Messages
(21)			: X	: : Menu
				: : Driven
(22)			: X	: : Additional
				: : Msg Info
(23)			: X	: : Portable

Validation Tests. Tests will be performed in assessing the validity of the MTS to satisfy the specifically identified message transfer system requirements. Validation tests will provide a validation that individual requirements are satisfied by the computer system program. Tests have an alpha and beta phase. Alpha testing is done at an AFIT laboratory. Beta testing is performed in the operational environment.

Expected Software Response. This section details the validation tests used to validate the MTS. Table II provides the requirement number, the associated validation test, and the associated expected software response. No attempt is made in this section to cover details relating to requirements that involve documentation, requirements 13 and 20. Documentation products will undergo a configuration audit during beta testing by PRN administrators and PRN operators.

Table II. Requirement, Validation Test, and Expected Software Response

<u>Req</u>	<u>Validation Test</u>	<u>Expected Software Response</u>
(1)	: Send a message to all : nodes.	: Msg received correctly : at all nodes.
(2)	: Place all message info, : such as author, date, : etc., into a message : along with text.	: Msg transferred to : 5 1/4-inch disk com- : plete with all addi- : tional information.

Table II. Requirement, Validation Test, and Expected Software Response (Cont'd)

Req	Validation Test	Expected Software Response
(3)	: Compose a message of : more than one and a half : pages of text.	: Only 1 1/2 pages of msg : text is transferred to : 5 1/4-inch disk.
(4)	: Send a message to a : known destination that : has no direct connec- : tion.	: Msg arrives at destina- : tion via an alternate : route.
(6)	: Send a message.	: Message arrives com- : plete.
(8)	: Continuously enter mes- : sages into the transmit : queue with a single : destination until a : Queue Full indication. : Count the number of : messages entered. Don't : immediately have the : destination operator : receive messages.	: The number of messages : entered will be equal : to or slightly greater : than TXQUEUEMAX, due to : transmit during entry. : If TXQUEUEMAX equals : RCVQUEUEMAX, then the : number of messages : available to the : receive operator is : RCVQUEUEMAX.
(9)	: Send a message to a : known off-line node.	: Indicators inform the : operator that the msg : did not get through : after TIMEOUT minutes : or TRYOUT tries.
(10)	: Fill the transmit : message queue with a : low-priority msg to a : single node. Immediately : send a msg to the same : node at a higher : priority. Enter no addi- : tional messages.	: Some of the low- : priority msgs get : through before the : higher priority msgs. : The remaining lower : priority msgs follow.
(11)	: Send a msg to a node : having an unattended : operator position.	: Msg is received at the : destination and placed : in the receive archive : file.

Table II. Requirement, Validation Test, and Expected Software Response (Cont'd)

Req	Validation Test	Expected Software Response
(13)	: One node is processing the entry of a message. : A second node is sending the first node messages. : : One node continuously processes msg entries with a second node as a destination. The second node is able to receive the messages.	: Messages transmitted to the first node processing a msg entry are received. : : The second node receives the messages.
(14)	: Activate operator interface Repeatedly call up the help option for each operator option.	: The operator interface is menu-driven. The help feature details all major operator options.
(15)	: Input a semicompleted msg header, i.e., leave out the destination, source, author, etc.	: Msg is detected as incomplete and causes an incomplete indication at the operator position.
(16)	: Continuously enter messages into the transmit message queue with a single destination until a Queue Full indication. Count the number of messages entered.	: All messages entered are received. Message receipt by the operator proceeds through the received message queue.
(20)	: Activate the Generate Message Program. Enter a message.	: The screen displays a msg template and a cursor query for input. The msg input is stored on 5 1/4-inch disk.
(21)	: Activate the Generate Message Program. Repeatedly call up the help feature for all customer options, and msg fields.	: The Generate Message Program is menu-driven with accurate, clearly worded, detailed help features.

Table II. Requirement, Validation Test, and Expected Software Response (Cont'd)

<u>Req</u>	<u>Validation Test</u>	<u>Expected Software Response</u>
(22)	: : Activate the Generate : Message Program. : : : : :	: : The Generate Message : Program has destina- : tion, author, date, : time, priority, : security, and message : text fields. :
(23)	: Activate the Generate : Message Program on an : IBM PC computer that : runs MS-DOS 3.X.	: The Generate Message : Program runs correctly. : :

### Conclusion

This chapter establishes the message transfer system requirements, identifies which of those requirements are satisfied by the MTS, and establishes the criteria that is used to validate the identified requirements satisfied by the MTS.

### III. System Design

#### Introduction

The goal of the design effort is to produce an operational MTS for the AFLC's PRN. Details of the system's objectives and major design constraints are presented as part of this introduction to the system's design. Following this, the four main sections of the system design chapter-- design description, system modules, error recovery and exception handling, and design verification--are developed.

System Objectives. The primary objective of the MTS is to transmit a message from any system node to any other system node over a shared HF channel. A direct radio propagation path between any two system nodes is not always available. An optional routing of the message through intermediate nodes must be provided. This optional routing is the purpose of the network routing algorithm.

Secondary objectives of the MTS involve satisfying the other ancillary requirements uncovered during the requirements analysis of Chapter II.

Along with specific system objectives are three overall objectives, first presented in Chapter I:

- (1) Portability--The MTS must be designed to operate with any IBM PC or compatible computer running

MS-DOS 3.0 or higher and having at least 512k bytes of RAM.

- (2) Flexibility--The MTS must have the ability to grow.
- (3) Maintainability--All system program elements must be designed to ease understanding, correcting, and enhancing.

These three overall objectives must be satisfied by all aspects of the MTS design.

Major Design Constraints. Major design constraints result from both hardware and software items that are part of the system's node at the time design begins. Another major design constraint results from the real-time nature of the MTS.

The major design constraints imposed by the system's hardware originate from two items, the terminal node controller and the microcomputer. The use of a specific terminal node controller, the PK-232, forces the use of a specific command and data exchange format on the computer-to-TNC interface (see Appendix A). The speed with which data can be exchanged between the computer and the TNC is limited by the capability of the PK-232. The system's use of the Zenith Model Z-150 microcomputer, an IBM PC-compatible computer equipped with 512k bytes of RAM memory and no hard disk drive, further narrows design implementation choices.

A major design constraint imposed by existing software is the system's use of the MS-DOS operating system. This is the operating system used on the system's microcomputers that must be used as the foundation of this software design.

The real-time nature of the MTS imposes design constraints by limiting the efficiency of program operation due to priority service interruptions.

### Design Description

The design description is a preliminary design step [18:256]. This part of the system development process relies on the data flow diagrams produced during requirements analysis. Study and refinement of these data-flow diagrams provide input to the two primary steps of design description, the data description and the program structure.

The overall design of the MTS can be viewed as being modeled by the seven-layer Open Systems Interconnection (OSI) model [23:15-21]. The bottom two layers are fully satisfied by the TNC. The top four layers are satisfied by the computer program. A traditional view of the third layer, the network layer, views the X.25 protocol as satisfying layer three responsibilities [23:238]. The TNC implements a version of X.25, AX.25 (see Appendix A). However, a closer look at how X.25 accomplishes the network

function reveals that it has no capability to perform network routing decisions [16]. The MTS computer program will have the capability to perform network routing decisions dynamically and is therefore viewed as satisfying the third layer of the OSI model.

Data Description. The purpose of a data description is to present the system data structures. Both physical and logical details for the two system data structures, the message data structure and the routing table data structure, are developed. The physical details associated with these data structures involves a discussion of the type of programming construct used to store and sort the data structures. The logical details involve an overview of how the data structures are used by programming processes.

Message Data Structure. Physically, the message data structure is composed of 12 fields, see Figure 7. Ten of the fields are alphanumeric, each having a fixed maximum number of characters. Two of the fields are pointers used by the programming construct that stores the message data structure. Each message data structure is stored as a separate entry. The principal purpose of the MTS is to transmit message data structures between system nodes.

source	destination	author	priority	security	date	subject	text	transmit time	recv time	pointer	pointer
--------	-------------	--------	----------	----------	------	---------	------	---------------	-----------	---------	---------

Figure 7. Representation of the Message Data Structure Broken Down by Fields

The message data structures are stored in linked lists. Linked lists are programming constructs that allow a fixed number of various types of data fields to be stored as a single data structure entry and data structure entries to be stored in sequence. The sequencing of data structure entries within the linked list is based on the characteristics of a specific data field in each data structure entry. A distinctive property of linked lists is that each linked list entry has fields that point to the prior linked list entry and the next linked list entry.

There are four linked lists in the program that involve the message data structure. The four linked lists are the generate message linked list, the transmit queue linked list, the transmitted message linked list, and the received message linked list.

A message data structure moves among different system linked lists. A message data structure is created during the generate message process. During program processing, a message data structure moves from the generate message linked list to the transmit queue linked list. After the message data structure has been transmitted, the message data structure is deleted from the transmit queue linked list and added to the transmitted message linked list. At the receiving node, the newly received message data structure is added to the received message linked list.

The sorting of message data structures within each of the four linked lists is done relative to only one field in each message data structure entry. Within the generate message linked list, the message data structure entries are sorted based on a message data structure's source field. For the transmit queue linked list, message data structure entries are sorted based on the message data structure's priority field. For the transmitted message linked list, message data structures are sorted based on a message data structure's destination field. For the received message

linked list, message data structures are sorted based on the message data structure's source field.

Logically, the message data structure is used by a number of program processes to make decisions. Typically, only a limited number of message data structure fields is used by any one program process. The destination field is used to determine a message's destination. The priority field is used to sort messages in the transmit queue linked list.

Routing Table Data Structure. Physically, the routing table data structure is composed of six fields, see Figure 8. Four of the fields are alphanumeric, each having a fixed maximum number of characters. Two of the fields are pointers used by the programming construct that stores the routing table data structure. Each routing table data structure is stored as a separate entry. The purpose of a routing table data structure entry is to supply the program with routing information to establish a communications link between source and destination.

The programming construct used to store the routing table data structures is a linked list. Each system has a routing table linked list with seven routing table data structure entries. Each of the routing table data structures represents a different destination. Each system node has seven possible destinations.

destination	route_1	route_2	route_3	pointer	pointer
-------------	---------	---------	---------	---------	---------

Figure 8. Representation of the Routing Table Data Structure Broken Down by Fields

The routing table linked list is used to determine the path for transmission of a message. The destination of the message is matched with a routing table linked list data structure. The routing paths associated with the destination are used as the message's transmission path. Alternate routing paths are provided for each destination so that if the best path cannot be used, other options are available. Updating of the routing path entries for each destination are accomplished by the routing algorithm. The routing algorithm dynamically determines the best routing paths and updates the routing table data structures.

Program Structure. The MTS program structure is developed based on the three software engineering concepts of an

established program hierarchy, modularity, and information hiding. [18:219-228].

The MTS is a real-time system. Real-time processing decisions cause the state of the system to change. At the foundation of the program structure is the idea of state-driven processing. A state transition manager is at the top of the program hierarchy. The state transition manager changes the system's state, depending on the system's condition. These changes of state are reflected as calls-to-program process modules by the state transition manager. Information hiding is supported by each process module only having access to the specific information needed during its processing.

The system structure chart was formed from the DFDs of Chapter II, see Figure 9. A transaction center is shown as the state transition manager. This transaction center makes decisions, based on the system's condition, of which process module to call [17:242-251]. These process modules are considered to be system states.

A review of the system structure chart is presented in the next section, followed by a discussion of changing between system states.

System Structure Chart. The system structure chart, Figure 9, shows the state transition manager at the top of the program hierarchy. The system structure chart

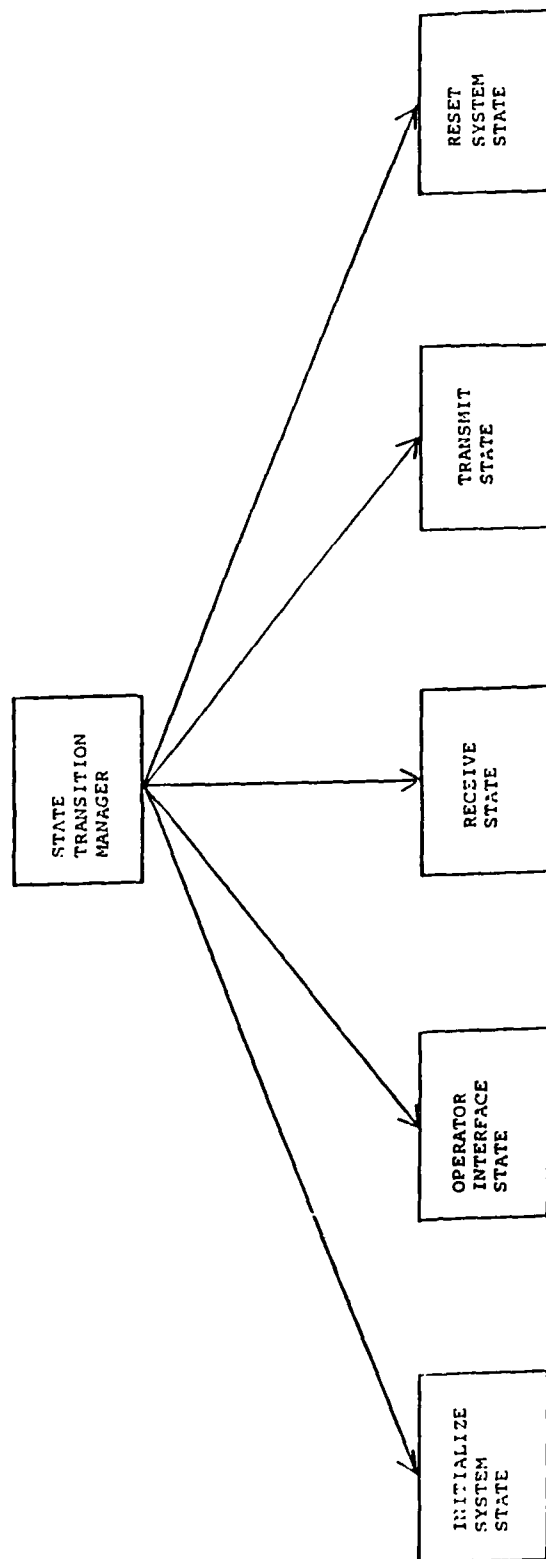


Figure 9. System Structure Chart

shows the two ancillary system states, the initialize system state and the reset system state, and the three fundamental system states, the operator interface state, the receive state, and the transmit state.

The initialize system state and the reset state are called ancillary modules because they are called only once during any active program period. The initialize system state is primarily responsible to initialize the communication link between the computer and the TNC and provide the TNC with station identifiers and system parameters. The reset system state returns the TNC to an idle condition, clears the communications link between the computer and the TNC, and stores the system linked lists.

The operator interface state is responsible for interfacing the MTS to the operator. This interfacing includes displaying menu choices on the computer's monitor and responding to requests for service. Once the menu choices are displayed and until there is a keyboard key depressed, the operator interface state turns program control back over to the state transition manager.

The receive state is responsible for receiving messages. The messages are stored for later access.

The transmit state is responsible for transmitting the highest priority message in the transmit queue linked list. Once a message is transmitted, it is stored for later reference.

The structure chart in Figure 9 defines the basic architectural structure of the MTS. This basic architectural structure displays an established program hierarchy and modularity. Information hiding is supported by each module having access only to information necessary to complete its process.

Changing System States. The state of the system is changed, based on system conditions. The system's three fundamental states are the operator interface state, the receive state, and the transmit state. Two ancillary states are the initialize system state and the reset system state. The three conditions that cause the system to change between fundamental system states are a receive request, a transmit request, and a keyboard key being depressed. The two conditions that cause the system to change to the two ancillary states are program initiation and an exit request.

Figure 10 illustrates changing between the three fundamental states. The highest priority of service is given to a receive request. Detecting a receive request causes the system to move to the receive state. Once a message has been received, the transmit queue is checked. If there is a message in the transmit queue and the transmit flag is enabled, the system moves to the transmit state. Once a message has been transmitted, there is a check mode for keyboard entries (kbhit). If a kbhit has occurred, the

system moves back to the operator interface state to process the keyboard request.

The two ancillary system states are active only once during each active session. The initialize system state is

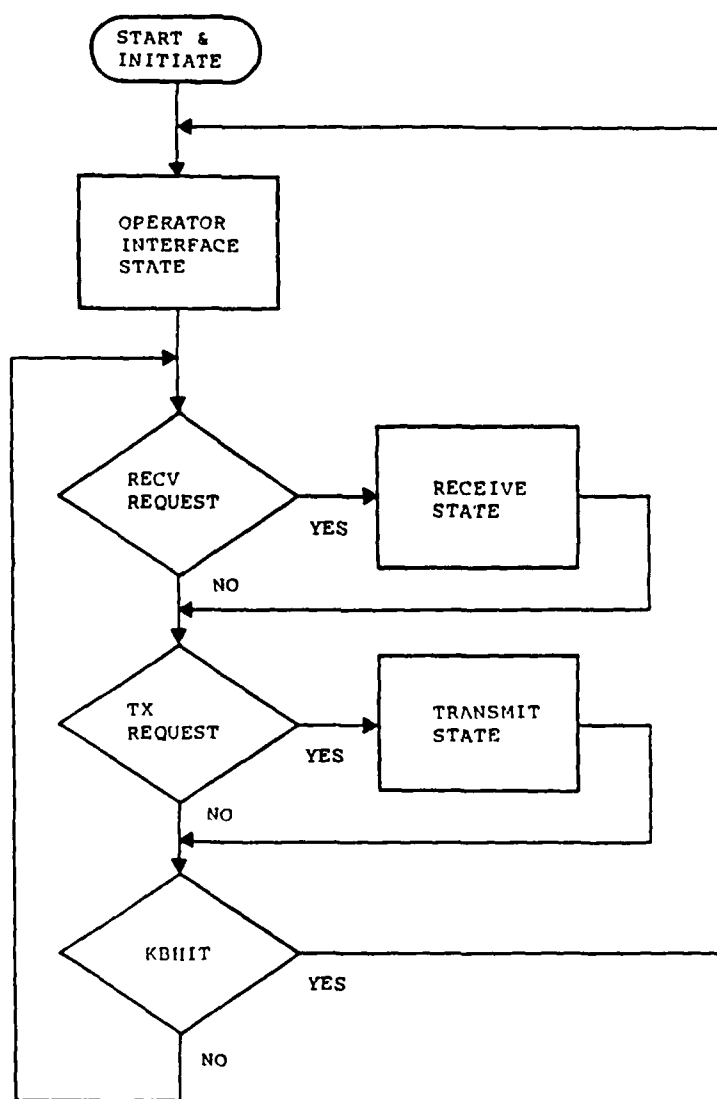


Figure 10. Flowchart Depicting the System State Change Process

called by the state transition manager when the MTS computer program is first initiated. The reset system state is

called by the state transition manager only upon an operator's exit request.

The previous discussion of changing the system state highlighted the system conditions that cause changes to the fundamental system states. A brief discussion of the system's two ancillary states was also presented.

### Modules

The system's modules include the state transition manager and the fundamental and ancillary states illustrated in the System Structure Chart of Figure 9. The system is composed of primary modules and submodules. The primary modules are the state transition manager, the initialize system state, the operator interface state, the receive state, the transmit state, and the reset state. There is a number of submodules associated with each of these primary modules that is responsible for portions of the primary module's processing. Structure charts, narrative discussions, and pseudocode descriptions for each of the modules and submodules are now presented.

State Transition Manager Module. The primary purpose of the state transition manager module is to change the system's state. Figure 9 is a structure chart for the state transition manager module showing the state modules that are

called by the state transition manager. Figure 10 illustrates the system conditions that are involved in making the decision of the next system state condition.

A pseudocode description depicts the processing of the state transition manager.

```
BEGIN STATE TRANSITION MANAGER
  INITIALIZE SYSTEM STATE(),
  WHILE(! EXIT)
    CHANGE TO OPERATOR INTERFACE STATE,
    WHILE(! KEY DEPRESSED)
      IF(RECV REQ = 1)          PROCESS A RECEIVE,
      IF(TRANSMIT REQ = 1)      PROCESS A TRANSMIT,
                              BACK TO OPERATOR
                              INTERFACE STATE
    END WHILE,
  RESET STATE(),
END STATE TRANSITION MANAGER
```

The code for the state transition manager is written in the C programming language (C) and presented in Appendix G.

None of the primary system data structures detailed previously is acted upon by the state transition module.

Initialize System State Module. The three responsibilities of the initialize system state module are to initialize the computer-to-TNC communications port, initialize the TNC, and initialize the system linked lists. Refer to Figure 11, the structure chart for the initialize system state. The module's responsibilities are performed by three submodules: the initialize port module, the initialize TNC module, and the initialize linked list module.

The pseudocode description of the initialize system state module shows that the only processing done is to call the three state's submodules.

```
BEGIN INITIALIZE SYSTEM STATE
  INITIALIZE PORT(PORT,PARAMETERS),
  INITIALIZE TNC(PORT),
  INITIALIZE LINKED LISTS(),
END INITIALIZE SYSTEM STATE
```

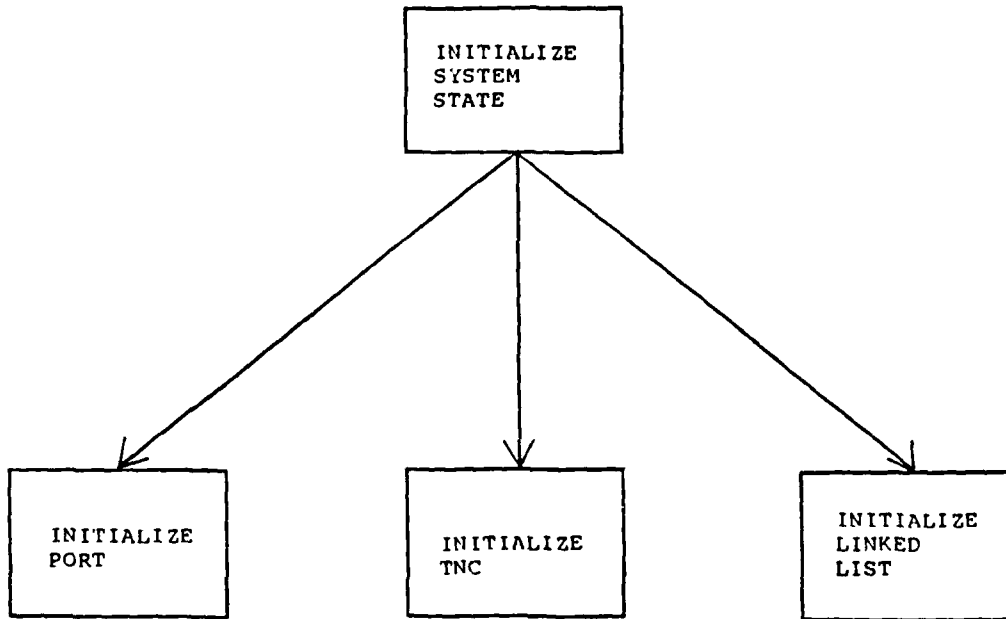


Figure 11. Initialize System State Structure Chart

The code for the initialize system state module is written in C and is presented in Appendix G.

The actions of the three submodules of the initialize system state module are now detailed.

Initialize Port Module. The initialize port module causes the computer's serial port to be set to specific

parameters. The parameters passed to the module specify the ports data transfer rate, parity, and the number of bits per word. Additionally, the module allocates a buffer for storing received data.

A pseudocode description of the initialize port module is:

```
BEGIN INITIALIZE PORT (PORT, PARAMETERS)
  ENABLE RECEIVE BUFFERS,
  SET PORT PARAMETERS,
END INITIALIZE PORT
```

The initialize port module's code is written in the assembly programming language and presented in Appendix G.

Initialize TNC Module. The initialize TNC module initiates the dialogue between the computer and the TNC. During the first steps in this dialogue, the TNC is instructed to move from start-up to fully operational status. This dialogue is accomplished through the use of send and receive commands. The send command transfers data out the computer's serial port to the TNC. The receive command reads data coming into the computer's serial port from the TNC. The send and receive commands are written in Assembly and presented in Appendix G. The structure chart associated with the initialize TNC module is presented in Appendix D.

The pseudocode description of the initialize TNC module is:

```

BEGIN INITIALIZE TNC (PORT)
  WHILE(INCORRECT RESPONSE)
    SEND(PORT,COMMAND DATA),
    RESPONSE = RECV(PORT),
  END WHILE
  WHILE(INCORRECT RESPONSE)
    SEND(PORT,COMMAND DATA),
    RESPONSE = RECV(PORT),
  END WHILE
  .
  .
  .
END INITIALIZE TNC

```

The number of while process loops within the pseudocode description is a function of the number of command data streams that are to be sent to the TNC. The number of command data streams is a function of the number of TNC parameters that must be set for a particular node's configuration. The code for the initialize TNC module is written in C and presented in Appendix G.

Initialize Linked Lists Module. This module is responsible for initializing the system's linked lists. There are four linked lists associated with each node: the transmit queue linked list, the transmitted messages linked list, the received messages linked list, and the routing table linked list. The structure chart for the initialize linked lists module is presented in Appendix C.

During previous system operation, system linked lists were stored on a 5 1/4-inch disk as the program was exited. The job of the initialize linked list module is threefold:

search the 5 1/4-inch disk for the previous linked lists, provide the operator with the choice of whether to use the previous linked lists or to start the linked list in the cleared state, and to perform the operator's choice with regard to the linked lists. In the event that no previous linked lists are recovered, no operator choice is possible and the system linked lists are set up in the cleared state.

A pseudocode description of the initialize linked lists module is:

```
BEGIN INITIALIZE LINKED LISTS
  IF(RETRIEVE PREVIOUS LINKED LISTS)
    QUERY OPERATOR; LOAD OLD OR LOAD CLEAR,
    IF(LOAD OLD) LOAD OLD LINKED LISTS,
    IF(CLEAR) LOAD LINKED LISTS CLEARED,
  ELSE LOAD LINKED LISTS CLEARED
END INITIALIZE LINKED LISTS
```

The modules code is written in C and is presented in Appendix G.

The initialize system state module and its three component submodules, initialize port, initialize TNC, and initialize linked lists have been detailed.

Operator Interface State. The operator interface state is responsible for providing an interface between the system operator and the MTS. The principal agents involved in this interfacing are the computer monitor, the computer keyboard, the computer disk drives, and the system printer.

The operator interface is menu-driven. Menu choices appear on the computer monitor. Selection of a menu item by

the operator causes the program to display additional menus or requests for specific keyboard input [6:66-79]. Upon selection of a menu choice or upon entering the requested keyboard input, the selected operation is performed. The computer monitor continuously displays information about the operation being performed.

The operator interface state module initially presents a main menu on the computer monitor offering the following four choices to the system operator:

- TRANSMIT MESSAGE
- GENERATE A MESSAGE
- ACCESS ARCHIVED MESSAGE FILE
- EXIT PROGRAM

There is an active help feature available to the operator any time the system is in the operator interface state module. This help feature is invoked by depressing the F1 function key. The help display window appears in the center of the computer monitor and presents relevant information.

The operator interface state module uses four submodules to aid in processing its responsibilities: the display menu and prompt module; the prepare a message for transmit module; the access archive module; and the generate message module. A structure chart of the operator interface state is presented in Figure 12.

A pseudocode description of the operator interface state is:

```

BEGIN OPERATOR INTERFACE SYSTEM STATE
WHILE(!EXIT)
  WHILE(NO SELECTION)
    SELECTION = DISPLAY MENU AND PROMPT(MAIN MENU),
  END WHILE
  SWITCH(SELECTION)
    CASE(TRANSMIT MESSAGE):  PREPARE MSG FOR TRANSMIT(),
                           BREAK,
    CASE(ACCESS ARCHIVED MSG FILE):
                           ACCESS ARCHIVE AND UPDATE(),
                           BREAK,
    CASE(GENERATE A MESSAGE): GENERATE MSG(),
                           BREAK,
    CASE(HELP):  HELP(),
               BREAK,
    CASE(EXIT):  RETURN,
  END SWITCH
END WHILE
END OPERATOR INTERFACE STATE

```

The four submodules of the operator interface state module, the display menu and prompt module, the prepare a message for transmit module, the access archive and update module, and the generate message module, are detailed.

Display Menu and Prompt Module. The display menu and prompt module displays menu choices on the operator's monitor. Within this module the OIS interacts with the state transmission manager as previously discussed. A structure chart of the display menu and prompt module is presented in Appendix D.

A pseudocode description for the display menu and prompt module is:

```

BEGIN DISPLAY MENU AND PROMPT (MENU)
  DISPLAY MENU CHOICES,
  INTERACT WITH THE STATE TRANSITION MANAGER,
  IF(SELECTION) RETURN SELECTION
END DISPLAY MENU AND PROMPT

```

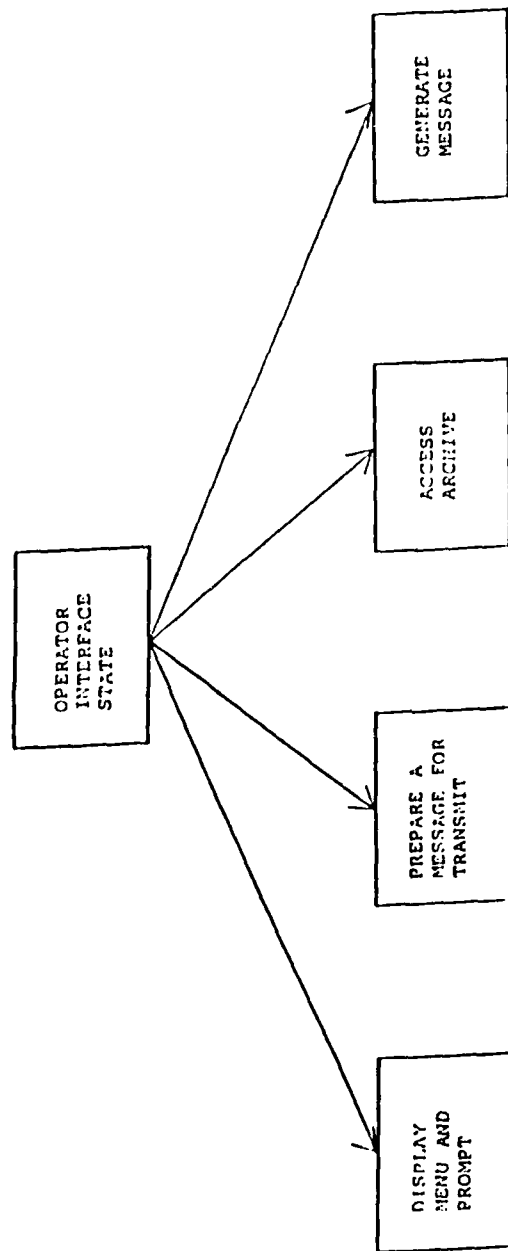


Figure 12. Operator Interface State Structure Chart

The code for the display menu and prompt module is written in C and presented in Appendix G.

Prepare a Message for Transmit Module. The prepare a message for transmit module is activated when the operator selects the TRANSMIT MESSAGE from among the main menu item choices. The module accomplishes the preliminary steps necessary before actually transmitting a message. This module does not transmit a message. The specific responsibilities of this module are to read a specified message from a 5 1/4-inch disk into RAM, check the message for completeness, reorder the transmit queue, and process any requests for the help function. The prepare a message for transmit module uses the display menu and prompt module to present the transmit menu options on the monitor. Once an operator makes a menu selection, the prepare a message for transmit module processes the selection. A structure chart of the prepare a message for transmit module is presented in Appendix C.

Upon entering the module, after having chosen TRANSMIT MESSAGE from the main menu, the transmit menu is displayed:

```
ENTER A GENERATED MESSAGE
BEGIN TO TRANSMIT
HOLDUP TRANSMISSION
RETURN TO MAIN MENU
```

Selecting the first transmit menu item causes the transmit submenu to be displayed:

ALL MESSAGES LISTED  
ONLY ONE MESSAGE LISTED  
ENTER TRANSMIT QUEUE LISTING  
RETURN TO MAIN MENU

A pseudocode description of the prepare a message for  
transmit module is:

```
BEGIN PREPARE A MESSAGE FOR TRANSMIT MODULE
  MOVE MSG FROM DISK TO RAM,
  CHECK MSG FOR COMPLETENESS,
  WHILE(!SELECTION)
    SELECTION=DISPLAY MENU AND PROMPT(TRANSMIT MENU),
  END WHILE
  SWITCH(SELECTION)
    CASE(ENTER A GENERATED MESSAGE):
      GO TO TRANSMIT SUBMENU
      RETURN
    CASE(BEGIN TO TRANSMIT):  ENABLE TRANSMIT FLAG,
      RETURN,
    CASE(HOLDUP TRANSMISSION):  DISABLE TRANSMIT FLAG,
      RETURN,
    CASE(RETURN TO MAIN MENU):  RETURN,
  END SWITCH
TRANSMIT SUBMENU:
  WHILE(!SELECTION)
    SELECTION=DISPLAY MENU AND PROMPT (TRANSMIT
      SUBMENU)
  END WHILE
  SWITCH(SELECTION)
    CASE(ALL MESSAGES LISTED):
      GET MESSAGES OFF DISK,
      ENTER MESSAGES INTO TRANSMIT
      QUEUE,
      REORDER TRANSMIT QUEUE,
      RETURN,
    CASE(ONLY ONE MESSAGE LISTED):
      GET MESSAGES OFF DISK,
      DISPLAY MESSAGES,
      WAIT FOR OPERATOR CHOICE,
      ENTER CHOSEN MESSAGE INTO
      TRANSMIT QUEUE,
      REORDER TRANSMIT QUEUE,
      RETURN,
```

```

CASE(ENTER TRANSMIT QUEUE LISTING):
    ALLOW OPERATOR TO ENTER
    MESSAGE,
    ENTER MESSAGE INTO TRANSMIT
    QUEUE,
    REORDER TRANSMIT QUEUE,
    RETURN
CASE(RETURN TO MAIN MENU):
    RETURN
END SWITCH
END TRANSMIT SUBMENU
END PREPARE A MESSAGE FOR TRANSMIT

```

The code for the prepare a message for transmit module is written in C and presented in Appendix G.

Access Archive Module. The access archive module enables the system operator to view the system linked lists. This module also gives the operator the opportunity to act upon the system linked lists. Initially, the module displays the archive menu, a choice of seven menu items on the operator's monitor:

```

DISPLAY ALL TRANSMITTED MESSAGE HEADERS
SHOW THE RECEIVED MESSAGE HEADERS
THE TRANSMIT QUEUE
A SPECIFIC NODE'S TRANSMIT AND RECEIVE MESSAGE HEADERS
MESSAGE HEADERS FROM A SPECIFIC TIME PERIOD
CURRENT ROUTING TABLE
RETURN TO MAIN MENU

```

When any one of the first five access archive menu items is selected, the monitor is cleared, then the requested linked list entries are displayed. Reverse video highlights one of the linked list entries. The linked list entry highlighted is selectable by the arrow keys. After the entry to

be processed is highlighted, the RETURN key is pressed and the archive submenu appears:

```
PRINT MESSAGE
STORE MESSAGE AS
VIEW MESSAGE
DELETE MESSAGE
LINKED LIST PRINT
GO TO ARCHIVE MENU
RETURN TO MAIN MENU
```

This menu partially pulls down over the linked listings. The linked list entry that was highlighted when the submenu was activated is still highlighted in reverse video. The up and down arrow keys are active, but now cause a reverse video highlight to cycle through the menu choices. Depressing the RETURN key causes the highlighted menu selection to process. Where this menu selection is for action on an individual linked list entry, such as print message, the highlighted linked list entry is acted upon. So, in this case, the message pointed to by the highlighted linked list entry is printed.

Referring back to the archive menu, when the sixth archive menu item is chosen, CURRENT ROUTING TABLE, the routing table linked list is displayed. When the correct routing table linked list entry is highlighted in reverse video, the RETURN key is depressed and the routing table menu appears on the monitor:

ADD A NEW ROUTING TABLE ENTRY  
MODIFY AN EXISTING ROUTING TABLE ENTRY  
VIEW ENTIRE ENTRY  
DELETE THE HIGHLIGHTED ROUTING TABLE ENTRY  
ENTER NEW ROUTING TABLE FROM DISK  
STORE CURRENT ROUTING TABLE ON DISK  
RETURN TO MAIN MENU

This menu partially pulls down over the linked listings. The linked list entry that was highlighted when the menu was activated is still highlighted in reverse video. The up and down arrow keys are active, but now cause a reverse video highlight to cycle through the menu choices. Depressing the RETURN key causes the highlighted menu selection to process. Where this menu selection is for action on an individual linked list entry, such as view entire entry, the highlighted linked list entry is acted upon. In this case, the routing table linked list entry highlighted is displayed on the monitor.

Once an archive submenu selection has finished processing, control is returned to the main menu.

The access archive module uses the display menu and prompt module to present the menu options on the monitor. Once an operator makes a menu choice, the access archive module processes the selection.

A pseudocode description of the access archive module is:

```

BEGIN ACCESS ARCHIVE
FOREVER,
ARCHIVE:WHILE(NO SELECTION)
    SELECTION=DISPLAY MENU & PROMPT(ARCHIVE MENU),
    END WHILE
    SWITCH(SELECTION)
        CASE(DISPLAY ALL TRANSMITTED MESSAGE HEADERS):
            LIST TRANSMITTED,
            PROCESS ARROW KEYS, RETURN KEYS, SPECIAL KEYS,
            UPON RETURN GO TO ARCHIVE SUBMENU,
            BREAK,
        CASE(SHOW THE RECEIVED MESSAGE HEADERS):
            LIST RECEIVE,
            PROCESS ARROW KEYS, RETURN KEYS, SPECIAL KEYS,
            UPON RETURN GO TO ARCHIVE SUBMENU,
            BREAK,
        CASE(TRANSMIT QUEUE);
            LIST TRANSMIT QUEUE,
            PROCESS ARROW KEYS, RETURN KEYS, SPECIAL KEYS,
            UPON RETURN GO TO ARCHIVE SUBMENU,
            BREAK,
        CASE(A SPECIFIC NODE'S TRANSMIT AND RECEIVE
            MESSAGE HEADERS):
            LIST SPECIFIC,
            PROCESS ARROW KEYS, RETURN KEYS, SPECIAL KEYS,
            UPON RETURN GO TO ARCHIVE SUBMENU,
            BREAK,
        CASE(MESSAGE HEADERS FROM A SPECIFIC TIME PERIOD):
            LIST TIME,
            PROCESS ARROW KEYS, RETURN KEYS, SPECIAL KEYS,
            UPON RETURN GO TO ARCHIVE SUBMENU,
            BREAK
        CASE(CURRENT ROUTING TABLE):
            LIST ROUTING,
            PROCESS ARROW KEYS, RETURN KEYS, SPECIAL KEYS,
            UPON RETURN GO TO ROUTING TABLE MENU,
            BREAK,
        CASE(RETURN TO MAIN MENU): RETURN,
    END SWITCH
END FOREVER
SUBMENU:SELECTION=DISPLAY MENU AND PROMPT (ARCHIVE SUBMENU),
    SWITCH(SELECTION)
        CASE(PRINT MESSAGE):
            PRINT MESSAGE,
            RETURN,
        CASE(STORE MESSAGE AS):
            STORE MESSAGE,
            RETURN,
        CASE(VIEW MESSAGE):

```

```

        VIEW MESSAGE,
        RETURN,
    CASE(DELETE MESSAGE):
        DELETE MESSAGE,
        RETURN,
    CASE(LINKED LIST PRINT):
        PRINT THE LINKED LIST,
        RETURN,
    CASE(GO TO ARCHIVE MENU):
        RETURN,
    CASE(RETURN TO MAIN MENU):
        RETURN,
    END SWITCH
END SUBMENU
ROUTING TABLE MENU:SELECTION=DISPLAY MENU AND PROMPT
                        (ROUTING TABLE SUBMENU)
    SWITCH(SELECTION)
        CASE(ADD A NEW ROUTING TABLE ENTRY):
            ADD ROUTING TABLE ENTRY,
            RETURN,
        CASE(MODIFY AN EXISTING ROUTING TABLE ENTRY):
            DISPLAY ROUTING TABLE ENTRY,
            MODIFY ROUTING TABLE ENTRY,
            RETURN,
        CASE(VIEW ENTIRE ENTRY):
            DISPLAY ROUTING TABLE ENTRY,
            RETURN,
        CASE(DELETE THE HIGHLIGHTED ROUTING TABLE ENTRY):
            DELETE ROUTING TABLE ENTRY,
            RETURN,
        CASE(ENTER NEW ROUTING TABLE FROM DISK):
            LOAD ROUTING TABLE ENTRIES FROM DISK,
            RETURN,
        CASE(STORE CURRENT ROUTING TABLE ON DISK):
            STORE CURRENT ROUTING TABLE ENTRIES ON DISK,
            RETURN,
        CASE(RETURN TO MAIN MENU):
            RETURN,
    END SWITCH
END ROUTING TABLE MENU
END ACCESS ARCHIVE

```

The code for the access archive module is written in C and presented in Appendix G.

Generate Message. The generate message module is responsible for generating a message. The module will also be available as a stand-alone program so that generation of messages can be accomplished off-line. The generate message option is available as part of the operator interface system to allow more flexibility.

The generate message module can be broken into three processes: display a message template on the monitor, prompt the operator through the message fields, and store the message on a 5 1/4-inch disk. There are three sub-modules that carry out these specific operations. A structure chart for the generate message module is presented in Appendix C.

A pseudocode description of the generate message module is:

```
BEGIN GENERATE MESSAGE
  DISPLAY MESSAGE TEMPLATE,
  PROMPT THRU TEMPLATE FIELDS,
  STORE MESSAGE,
END GENERATE MESSAGE
```

The code for the generate message module is written in C and presented in Appendix G.

The OIS module and its four component submodules--the display menu and prompt module, the prepare a message for transmit module, the access archive module and the generate message module--have been detailed.

Receive State Module. The receive state module is responsible for processing the receiving of messages over the computer-to-TNC communications link. Processing received messages involves three steps: the message must be received from the TNC, the path taken by the received message is used to update the routing tables, and the received message is added to the received message linked list. Processing received messages is done by three sub-modules: the receive message module, the update routing table module, and the update receive message archive module. Figure 13 is a structure chart of the receive state module.

A pseudocode description of the receive state module is:

```
BEGIN RECEIVE STATE
    RECEIVE MESSAGE(),
    UPDATE ROUTING TABLES(),
    UPDATE RECEIVE MESSAGE ARCHIVE(),
END RECEIVE STATE
```

The code for the receive state module is written in C and presented in Appendix G.

Each of the three submodules of the receive state module is now detailed.

Receive Message Module. The receive message module is responsible for receiving a message from the TNC. Receiving a message from the TNC is a five-step process that queries the TNC for data, interprets responses from the TNC,

concatenates received data to a received message data structure, and detects an end of message. A structure chart of the receive message module is presented in Appendix C.

A pseudocode description of the receive message module is:

```
BEGIN RECEIVE MESSAGE
  WHILE(1 EOM)
    QUERY TNC FOR RECEIVE MESSAGE DATA,
    INTERPRET RESPONSES FROM TNC,
    CONCATENATE RECEIVED DATA TO A MESSAGE DATA
                                     STRUCTURE,
  END WHILE
  PROCESS EOM,
END RECEIVE MESSAGE
```

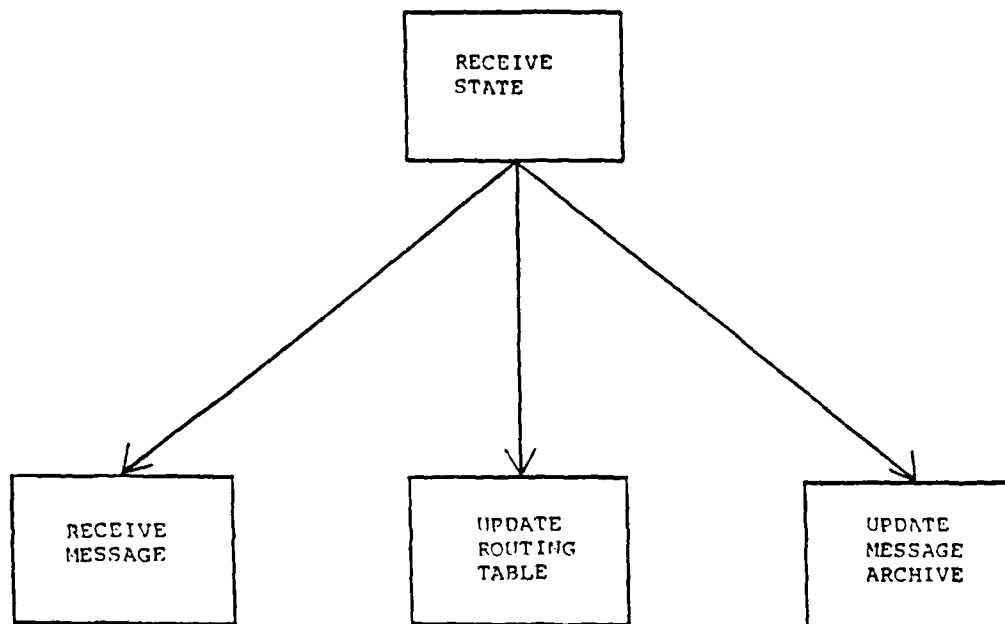


Figure 13. Receive State Structure Chart

The code for the receive message module is written in C and presented in Appendix G.

Updating Routing Table Module (Receive). The update routing table module is responsible for modifying the routing table linked list. This modification is done based on the transmission path taken by the recently received message. A structure chart of the update routing table (receive) module is presented in Appendix C.

The routing table linked list provides up to three transmission paths between a source and destination node. The top transmission path entry is considered to be the best route. The next transmission path in the listing is considered to be the second best path, and the third transmission path in the listing is considered to be the least desirable path.

Modifications to the routing table linked listing involve reordering the routing table linked list transmission paths. This reordering can also include adding a new path if the current received message transmission path is not in the choice of transmission paths. The recently received message's transmission path is considered to be the best current path between source and destination nodes.

The modification process searches for a match between the recently received message's transmission path and the

current routing table linked list transmission path entries for the destination. Should a match occur, the routing table linked list path entry is deleted. Irrespective of the matching outcome, the newly received message's transmission path is placed as the top choice of transmission paths in the routing table linked list. Other entries are placed below this entry and kept in their former sequence.

If the modification process starts with three transmission path choices, then at the end of the modification process there will still be three choices, even if the recently received message's transmission path was a new addition. From this starting condition and a new addition, the least desirable transmission path is deleted from the routing table linked list.

If the modification process starts with fewer than three transmission path choices, then a recently received message's transmission path that does not already appear as a transmission path choice is added.

A pseudocode description of the update routing table (receive) is:

```
BEGIN UPDATE ROUTING TABLE (RECEIVE)
  SEARCH FOR A MATCH,
  REORDER PATHS,
END UPDATE ROUTING TABLE (RECEIVE)
```

The code for the update routing table (receive) module is written in C and presented in Appendix G.

Update Received Message Archive Module. The update received message archive module is responsible for changing the received message linked list to reflect the recently received message.

A pseudocode description of the update received message archive module is:

```
BEGIN UPDATE RECEIVED MESSAGE ARCHIVE
  ADD MESSAGE DATA STRUCTURE TO RECEIVED MESSAGE LINKED
LIST,
END UPDATE RECEIVED MESSAGE ARCHIVE
```

The code for the update received message archive module is written in C and presented in Appendix G.

The receive state module and its three component sub-modules, receive message module, update routing table module, and update received message archive module, have been detailed.

Transmit State Module. The transmit state module is responsible for processing the transmission of a message over the computer to TNC communications link. Processing the transmission of a message involves three steps: the transfer of the message from the computer to the TNC, updating the routing table linked list after transmission of the message, and updating the transmit queue and the transmitted message linked lists. These processes are handled by three submodules of the transmit state module. The three submodules are the transfer message module, the

update routing table module, and the update transmit message archive module. Figure 14 is a structure chart for the transmit state module.

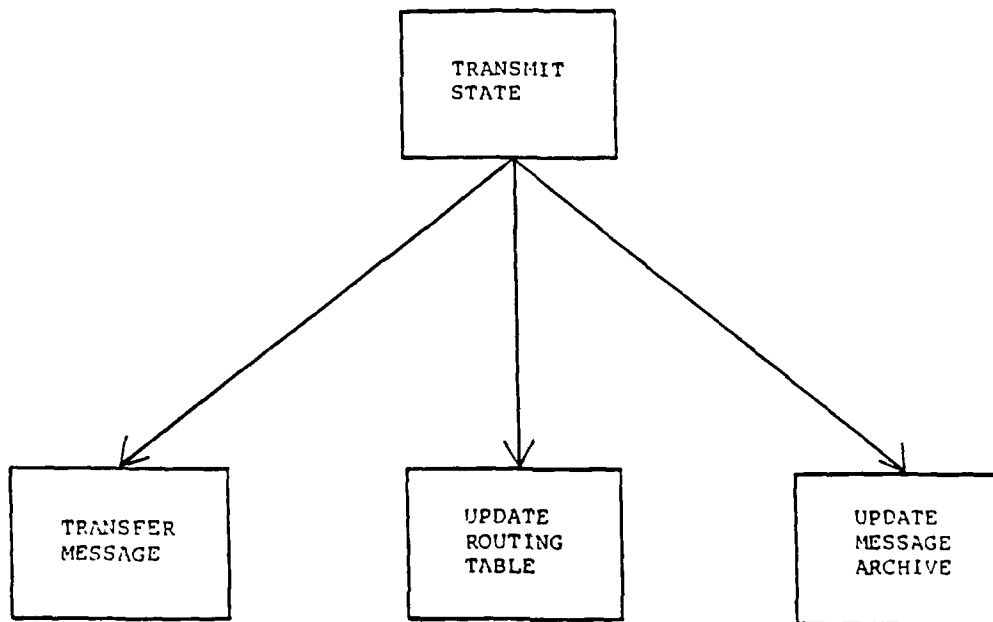


Figure 14. Transmit State Structure Chart

A pseudocode description of the transmit state module is:

```
BEGIN TRANSMIT STATE
  TRANSFER MESSAGE(),
  UPDATE ROUTING TABLE(),
  UPDATE TRANSMITTED MESSAGE ARCHIVE(),
END TRANSMIT STATE
```

The code for this module is written in C and presented in Appendix G.

Each of the three submodules of the transmit state module is now detailed.

Transmit Message Module. The transmit message module is responsible for transmitting a message. Transmission of a message is a three-step process that establishes a connection between the local and destination TNCs, transfers the message to the destination TNC, and orders a disconnect with the destination TNC.

A pseudocode description of the transfer message module is:

```
BEGIN TRANSFER MESSAGE
  CONNECT
  SEND BYTES,
  DISCONNECT,
END TRANSFER MESSAGE
```

The code for the transfer message module is written in C and presented in Appendix G.

Update Routing Table Module (Transmit). The update routing table module (transmit) is responsible for modifying the routing table linked list. This modification is done based on the transmission path taken by the recently transmitted message. A structure chart of the update routing table (transmit) module is presented in Appendix C.

The recently transmitted message's transmission path is considered the current best transmission path to the destination node. The recently transmitted message's transmission path comes directly from the routing table linked list. This transmission path is used to modify the linked list.

The modification process searches for a match between the recently transmitted message's transmission path and the current routing table linked list transmission path entries. If there is a match with the first transmission path, the modification process is concluded, the best current path is at the top of the list. If a match is not made with the first transmission path entry, the newly transmitted message's transmission path appears somewhere else in the listing of the three transmission path choices. It is found and deleted from the linked list. The newly transmitted message's transmission path replaces the top entry, the former top entry falls into the second-best entry position. The other entry is then moved to the third-best entry position.

At the end of the modification process, the routing table linked list contains the same number of transmission path choices as when the modification process began. The transmit function does not generate any new transmission paths. The only purpose of the modification process is to reorder the routing table linked list in the event that the top transmission path was not successful at providing a usable transmission path.

A pseudocode description of the update routing table (transmit) module is:

```

BEGIN UPDATE ROUTING TABLE (TRANSMIT)
  SEARCH FOR A MATCH,
  REORDER PATHS,
END UPDATE ROUTING TABLE (TRANSMIT)

```

The code for the update routing table (transmit) module is written in C and presented in Appendix G.

Update Transmitted Message Archive Module. The update transmitted message archive module is responsible for changing the transmitted message linked list to reflect the transmission of a new message. Additionally, the module revises the transmit queue linked list by deleting the entry associated with the recently transmitted message. The transmit queue linked list is then reordered by priority. A structure chart of the update transmitted message archive module is presented in Appendix C.

A pseudocode description of the update transmitted archive module is:

```

BEGIN UPDATE TRANSMITTED MESSAGE ARCHIVE MODULE
  MOVE TRANSMITTED MESSAGE TO TRANSMITTED MESSAGE
                                     LINKED LIST,
  REVISE TRANSMIT QUEUE,
END UPDATE TRANSMITTED MESSAGE ARCHIVE MODULE

```

The code for the update transmitted message archive module is written in C and presented in Appendix G.

The transmit state module and its three component sub-modules, transmit message module, update routing table module, and update transmitted message archive module, have been detailed.

Reset System State Module. The three responsibilities of the reset system state module are to reset the TNC, close the computer-to-TNC communications port, and store the system linked lists. These responsibilities are performed by three submodules, the reset TNC module, the close port module, and the store linked lists module. Figure 15 is a structure chart of the reset system state module.

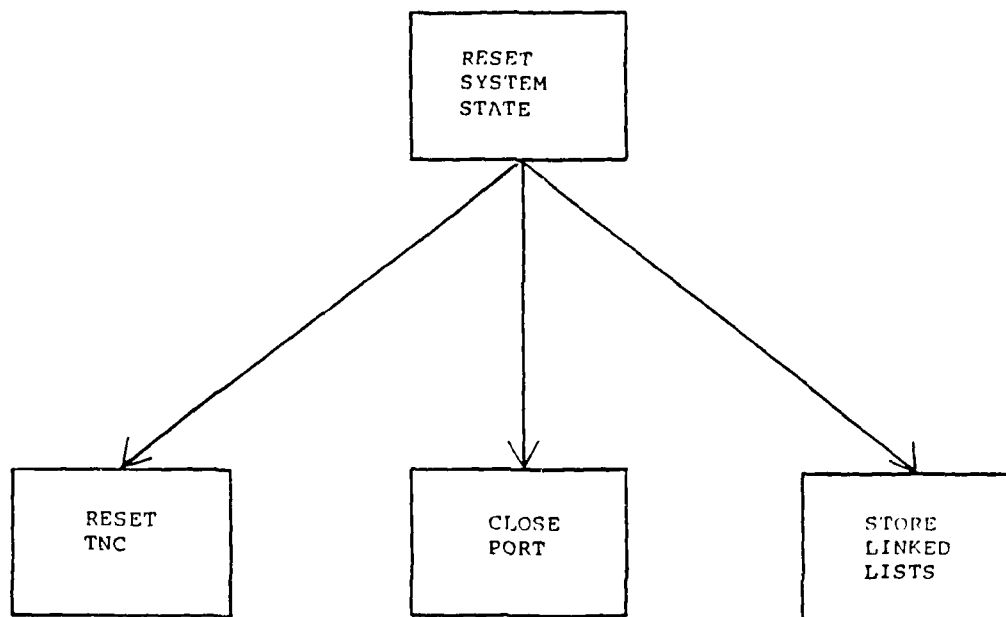


Figure 15. Reset System State Structure Chart

A pseudocode description of the reset system state module is:

```

BEGIN RESET SYSTEM STATE
  RESET TNC(),
  CLOSE PORT(),
  STORE LINKED LISTS(),
END RESET SYSTEM STATE

```

The code for the reset system state module is written in C and presented in Appendix G.

Each of the three submodules of the reset system state module is now detailed.

Reset TNC Module. The reset TNC module controls a dialogue between the computer and the TNC. During this dialogue, the TNC is issued commands which instruct it to move from the fully operational state to a reset state.

A pseudocode description of the reset TNC module is:

```

BEGIN RESET TNC
  SEND(PORT, COMMAND DATA),
  RECV(PORT, COMMAND RESPONSES),
  SEND(PORT, COMMAND DATA),
  RECV(PORT, COMMAND RESPONSES),
  .
  .
  .
END RESET TNC

```

The code for the reset TNC module is written in C and is presented in Appendix G.

Close Port Module. The close port module causes the computer-to-TNC communication port to be closed. The module's code is written in the Assembly programming language and is presented in Appendix G.

Store Linked Lists Module. The store linked lists module is responsible for storing the four system linked lists on a 5 1/4-inch disk at the time reset is initiated.

The code for the store linked lists module is written in C and is presented in Appendix G.

The reset state module and its three component modules, reset TNC module, close port module, and store linked lists module, have been detailed.

#### Error Recovery and Exception Handling

The topic of error recovery and exception handling covers situations where the normal flow of processing by the computer program is interrupted or disabled by some system-triggered event. An example is when a process to store data on a 5 1/4-inch disk cannot open the file. How this and similar situations are handled by the computer program are of importance because improper error recovery or exception handling can cause the MTS to hang up or even to crash.

The topic of error recovery and exception handling is broken into two areas for detailed discussion--error recovery and exception handling.

Error Recovery. Errors related to this discussion center around three system components: the keyboard, the 5 1/4-inch disk, and the TNC. Each component requires a different error recovery process.

Keyboard errors could be caused by invalid keys being entered by the operator. During an active display on the computer monitor, only certain keys are valid input. Specifically, when the menu choices are displayed, the only valid keys are the up and down arrow keys, the enter key, and the character keys of the first letter of any menu choice. Error recovery for a keyboard error, hitting an invalid key, is performed by a masking operation recognizing only valid keys. Invalid keys are ignored.

The 5 1/4-inch disk could cause errors in three instances: failure to find a drive active, failure to find the file named, or failure to open a file, and a disk full error message. Error recovery in each of these instances involves an indicator message to be displayed on the computer monitor. The operator is then expected to perform normal recovery procedures, such as activating the disk drive, renaming the file, or inserting a 5 1/4-inch disk with additional storage capacity.

Errors associated with the TNC involve responses to computer commands and computer queries of the TNC that are unintelligible. Errors could occur while the computer is establishing TNC parameters, processing the transmission of a message, processing the receipt of a message, or resetting the TNC. The process for handling this type of error is to repeat the command or query and reevaluate the TNC's

response. This looping process occurs only a limited number of times until the operator is notified that there is a problem with the computer-to-TNC communication link. This notification is in the form of a monitor message stating the problem.

Exception Handling. Exception conditions involve processes that give information that is not expected; the information is intelligible. One such situation occurs when the TNC is queried by the computer for current message data. Instead of message data being received, other TNC information is received. This other information is, typically, information about radio channel traffic. Because the radio channel is shared among all the system's nodes, various pairs of nodes can be communicating over the radio channel. The additional information is about other pairs of system node communications that were heard by the local TNC. This information is called monitored data. The level of monitoring done by the TNC is selected so that there is no monitoring except for local node processing.

Exception handling is also necessary during transmitting and receiving a message. During this time, the entire message may not completely transfer over the radio channel due to the loss of the propagation path. Handshaking and interval timing are two ways that this exception condition is

handled. The handshaking occurring at the message transmission level involves a search by the receiving station for a specific character sequence that is sent by the transmitting node to indicate the end of a message. The receiving node then sends a specific character sequence to the transmitting node. Only after this handshaking at the end of a message's transmission will both nodes register the message transmission as successful. This prevents a channel outage resulting in an incorrect message transmission. Interval timing is also used by the MTS computer program to prevent a channel outage from tying up a node during transmission by having the node continuing to wait for an end-of-message indicator. After each packet is received by the MTS computer program, a timer is started. If the time before another packet is received exceeds the time limit, then the message transfer process is aborted and a disconnect is processed. The timer's limit is set by the system administrator. The exception condition of a radio channel outage is handled by both handshaking and interval timing.

The previous discussion of error recovery and exception handling detailed how these elements of program flow are handled by specific code segments placed at the point of possible occurrence of errors and exception conditions.

### Design Verification

Design verification answers the question, "Are we designing the product right?" During design verification, the system's requirements listing is matched to design components. All requirements that were to be satisfied by the MTS are listed. This matching of requirements to design components is presented in tabular form in Table III. The requirements from Chapter II that were to be satisfied by the MTS are listed in numerical order. The results presented in Table III show that the design satisfies the requirements.

Table III. Design Verification Table

## Requirement

	1	2	3	4	6	8	9	10	11	12	13	14	15	16	17	19	20	21	22	23
State Transition Manager	X										X									
Initialize System State																				
Initialize Port																				
Initialize TNC																				
Initialize Linked Lists																				
Operator Interface State	X					X	X	X			X	X	X	X	X					
Display Menu and Prompt	X											X								
Prepare Message for Transmit	X					X		X				X	X	X						
Access Archive					X							X			X					
Generate Message		X	X									X					X	X	X	
Receive State	X								X		X									
Receive Message	X								X											
Update Routing Table (Receive)				X																
Update Received Message Archive					X				X											
Transmit State	X					X	X	X			X									
Transfer Message	X						X													
Update Routing Table (Transmit)				X																
Update Transmitted Message Archive					X	X		X												
Reset System State																				
Reset TNC																				
Close Port																				
Store Linked Lists																				

## IV. Testing

### Introduction

Testing affords a view of the development process unrevealed by simply looking at the final system design. The design shows a static view of the finished product. Testing shows the dynamic development process that leads to the finished product. This dynamic development is particularly evident during verification testing.

Verification testing and validation testing are the two stages in the testing process. Verification testing answers the question, "Are we building the product right?" Validation testing answers the question, "Are we building the right product?" [18:499]

The testing process for each testing stage is composed of a test configuration and testing procedures. The test configuration details the hardware and software requirements necessary to perform the testing. The testing procedures list, in outline form, the sequence of actions taken by the tester and the expected program responses. A successful test is one where the expected program responses occur.

Specific details associated with verification and validation testing are presented, followed by a chapter summary.

## Verification Testing

Verification testing is composed of unit testing and integration testing. Unit testing looks at an individual module's function. Integration testing addresses the interaction between various modules.

Verification testing is started at the earliest stages in the software coding process and finishes when the finished software product is declared ready for validation testing.

Both unit testing and integration testing make use of embedded test code that displays program progress status messages. These program progress status messages are path-dependent.

A presentation of unit testing is followed by a presentation of integration testing.

Unit Testing. Unit testing looks inside the module to determine whether the module's process is being carried out correctly [13:679-681]. In this regard, unit testing can be seen to be a white-box testing method.

Unit testing makes extensive use of drivers and stubs. Drivers provide a vehicle to invoke the module under test. Drivers emulate the module-calling process required by the system design. Stubs provide the module under test with the

responses to function calls made by the module under test. These responses emulate the responses called for by system design.

The four principal test areas involved in unit testing are the examination of the data structures that are processed by the module under test, the boundary conditions associated with the module under test, independent paths within the module under test, and error handling paths within the module under test. A description of each of these four unit test areas is presented.

Data structure testing evaluates the characteristics associated with the data items handled by the computer program. For example, if alphanumeric characters are permissible, numeric digits should also work. If a data string is able to be five alphanumeric characters long, are five alphanumeric characters able to be put in the data string? Did the symbol of the key that was depressed become the symbol stored?

Boundary condition testing evaluates any limits built into the computer program. For example, if five characters can be stored in a data string, what happens when an attempt is made to store no characters, or an attempt is made to store six characters?

Independent path testing looks at a program's flow through the module. Typically, these independent paths

follow a transaction center. A transaction center is where one or more system conditions are evaluated and the resulting decision causes a specific program path. Testing of independent paths involves setting up the conditions to process through a specific path and seeing that that path is chosen. The goal in this type of testing is to check all the module's independent paths.

Error handling path testing ties together the results of post-processing of a transaction decision point after a boundary condition has been violated. The resulting actions constitute an error handling path. Error handling paths can also result from a number of other sources, such as improper responses to a function call, or system-constraining items, such as being out of memory.

The principal method for implementing these four types of tests is through the use of embedded routines that cause program progress status messages to be displayed on the monitor, depending on the module's internal execution sequence. The appearance of the proper program progress status messages in the proper sequence indicates the program's internal execution is correct.

There are five foundation modules that will undergo unit testing. These five foundation modules provide a base for development of the system modules. The presentation of each

of the five foundation modules details the module's function and the specific areas to be examined during testing.

The display menu and prompt module

- display the menu listing passed in function parameter
- look for a keyboard hit and process: if selection, return; if arrow, move cursor; if valid letter, move cursor.
- look for a receive request: if receive request, go to receive
- look for a transmit request: if transmit request, go to transmit

Two specific test areas are examined during unit testing of this module: an examination of independent paths through the module, and an examination of error handling paths through the module.

The serial port connectivity module

- provides an ability to initialize the serial communications port
- provides an ability to send a data character out the serial communications port
- provides an ability to receive data characters automatically from the serial communications port and place in a buffer
- provides an ability to read a data character from the receive buffer
- provides an ability to close the communications port

Two specific test areas are examined during unit testing of this module: an examination of data structures processed by the module, and an examination of independent paths through the module.

The transmit message module

- queries the operator for a TNC connect path
- switches the TNC into HOST mode (see Appendix A)

- while in HOST mode, transfers a specified connect path to TNC
- while in HOST mode, looks for an active connect status from TNC
- queries the operator for a character string to transmit
- while in HOST mode, transfers a specified character string to the TNC
- while in HOST mode, provides a disconnect request to the TNC
- while in HOST mode, looks for a disconnect acknowledgement from the TNC

Three specific test areas are examined during unit testing of this module: an examination of data structures processed by the module, an examination of independent paths through the module, and an examination of error handling paths through the module.

#### The receive message module

- switches the TNC into HOST mode (see Appendix A)
- while in HOST mode, looks for a receive request from the TNC
- while in HOST mode, and an active receive request condition exists, takes in the character data from the receive buffer being sent by the TNC
- while in HOST mode, looks for a disconnect from the TNC
- while in HOST mode, displays the received message on the monitor

Three specific test areas are examined during unit testing of this module: an examination of data structures processed by the module, an examination of independent paths through the module, and an examination of error-handling paths through the module.

The linked list operations module

- creates a doubly linked list
- provides for numerous fields within each entry
- displays the linked list on the monitor
- provides for adding entries to the linked list
- provides for deleting entries from the linked list
- provides for sorting the entries of the linked list for a specified field
- provides for finding a referenced entry within the linked list and displaying on the monitor
- provides for storing the linked list on disk
- provides for loading a linked list from disk

Three specific test areas are examined during unit testing of this module: an examination of data structures processed by the module, an examination of boundary conditions of the module, and an examination of error handling paths through the module.

Appendix E, Test Plan, documents the specifics of the unit testing process to include the test configuration and the test procedure associated with each foundation module.

Integration Testing. Integration testing looks at the operations occurring between modules to determine whether a program is operating correctly. Because the individual modules involved in the testing are viewed as black boxes, integration testing can be thought of as black box testing [13:681-684].

Integration makes extensive use of drivers. Stubs are used infrequently during integration testing because of the

nature of the bottom-up integration process. In the bottom-up integration process, modules at the lowest level are tied together in clusters. These clusters are first joined together based on a common function. Later clustering is done at a higher level in the program hierarchy. Because clustering starts at the lowest level in the program's hierarchy, little use of stubs is necessary.

There are three primary clusters associated with this program's integration process. These clusters are the operator interface cluster, the communications cluster, and the linked list cluster. The operator interface cluster is built from the display menu and prompt module with added functionality to perform as a stand-alone backbone for the program. The communications cluster is built around the serial communications module, the receive message module, and the transmit message module. The communications cluster is developed to provide an intermediate communications ability enabling systematic development of the most critical program function, communication between computers. The linked list cluster is built from the linked list module and provides for development of all the linked lists used by the program, and all operations needed to be performed on the linked lists.

During integration, the three primary clusters are built first. This is followed by combining clusters in pairs and

finally by joining all three clusters to form the system program. In order for integration testing to be performed at each phase of this integration process, integration testing is accomplished in six phases. The six phases of integration testing are: operator interface cluster testing; communications cluster testing; linked list cluster testing; testing on the tying together of the operator interface cluster and the communications cluster; testing of the tying together of the communications cluster and the linked list cluster; and testing of the tying together of the operator interface cluster, the communications cluster, and the linked list cluster.

The primary testing done during integration testing involves the interfaces between the component modules. The primary method used for that testing will be equivalence testing [15:44-50]. Equivalence testing treats the component modules as black boxes, breaking up the elements of interface processing into valid and invalid systems. An important aspect of integration testing is testing the function parameters passed during function calls and return calls. The coding language C has a provision for checking these parameters through the use of function prototypes. Extensive use of function prototypes enables the C compiler to check parameter passing features.

Appendix E, Test Plan, documents the specifics of the integration testing process.

#### Validation Testing

Validation tests, answering the question, "Are we building the product right?" were presented in Chapter II, Requirements. The validation tests are listed and referenced to the underlying system requirement that is satisfied for a correct response to the validation test.

Validation testing is composed of two stages, alpha and beta testing. With each, the validation tests are used to measure the system's capabilities. Although similar tests are run during alpha and beta testing, the environment in which the two types of tests are run is different.

Alpha testing is done in an electronics laboratory. A test bed is set up to replicate the target system's operation. The test bed will consist of a three-node network, with two nodes using the target system type of TNC. The third node, which acts as an intermediate node, does not require a target system type of TNC, but one that acts similarly during operation as an intermediate node. Once the test bed is set up, validation tests are performed. The testing is performed by an individual familiar with operational PRNs. Upon satisfactory completion of alpha testing, the validation tests will be performed in a beta test.

Beta testing is done in the operational environment. This testing will be staged so that a gradual evolution to the complete network is obtained. This staging will begin with only two operation nodes. Upon successful validation testing at this level, a third node will be added. This process of stepwise node addition will continue until all eight network nodes are able to successfully run the validation tests. At this point, validation testing is complete and the system is ready for use by the end user.

#### Summary

Details of the testing process have been presented. The testing process is shown to consist of a set of verification and validation tests. The Test Plan presented in Appendix E, and the validation tests from Chapter II were referenced to be the documents used during testing. Results of these tests are presented in the following chapter.

## V. Results, Conclusions, and Recommendations

### Introduction

Three subjects are covered in this chapter. The first subject discusses the test results performed on the MTS computer program. The second subject presented is the conclusions arrived at after the process of design and development of the MTS computer program. The third subject presented is a recommendation for follow-on efforts related to this thesis.

### Test Results

Test results were arrived at after performing the tests presented in Chapter IV and detailed in Appendix E, Test Plan. The test results section is broken up into verification and validation testing. After a brief discussion about each section, the reader is referred to Appendix F, Test Results, a listing of actions performed and the program's response.

Verification Tests. The specific verification tests specified in Chapter IV were performed. These tests were broken into unit tests and integration tests. The results are detailed in Appendix F, Test Results. Although the results appear to process smoothly through the procedure, the actual testing included a large amount of program code

debugging. The results detailed show only the results performed on the debugged program.

Validation Tests. The specific validation tests specified in Chapter II were performed. The results are presented in Appendix F, Test Results. System requirements not satisfied by the program are noted. The beta test phase of validation testing was not performed.

### Conclusions

Validation testing pointed to the specific areas of the system requirements that were and were not satisfied by the current version of the MTS computer program. The areas that were coded aimed at building a solid foundation from which further coding could easily satisfy all system requirements. These include such areas as the code for the dialogue between the computer and the TNC, code to allow the operator to control the system, and code to generate, transmit, and receive a message between two nodes. Requirements not satisfied by code include the checking of a message for completeness, and the help feature.

In general, the scope of the project was very broad; any one of the areas of design, development, and testing could serve as suitable thesis topics. This thesis effort fully designed and partially developed the system to include actual coding, and performed tests on the portion of the

code that was developed. In this way, insight was gained into all areas of system development.

This thesis allowed application of the basic engineering principles of defining the system requirements, establishing levels of performance for the system, designing the method to be used to satisfy the requirements, developing the detailed design into a working product, and testing the working product against the expected level of performance.

#### Recommendations for Further Study

Three areas are recommended for further study. The first is to completely develop and test all system requirements. This also involves refining the system to provide all the accepted industry practices such as templating all forms that appear on the monitor, and providing clear and easy-to-use help features.

The second area recommended for further study is to perform beta testing of the MTS computer program on AFLC's PRN. Incremental implementation on the network could be pursued. Two of the network's system nodes are in the immediate geographical area, providing a basic foundation for incremental implementation.

The third area recommended for further study is the interconnectivity of AFLC's PRN with other DOD communication networks. Specifically, the capability for interconnection to DOD TCP/IP networks should be pursued. The TCP/IP standard is used on the Defense Data Network.

## Bibliography

1. Advanced Electronic Applications, Inc. Operating Manual Model PK-232 Data Controller. Lynwood, WA: Advanced Electronic Applications, Inc., 1987.
2. Advanced Electronic Applications, Inc. Technical Reference Manual Model PK-232 Data Controller. Lynwood, WA: Advanced Electronic Applications, Inc., 1987.
3. Air Force Logistics Command. Air Force Logistics Command's Survival, Recovery, and Reconstitution Plan. No. 55.
4. Campbell, Joe. C Programmer's Guide to Serial Communications. Indianapolis, IN: Howard W. Sams & Company, 1987.
5. Chesley, Harry R., and Mitchell Waite. Supercharging C with Assembly Language. Reading, MA: Addison-Wesley Publishing Company, 1987.
6. Dumas, Joseph S. Designing User Interfaces for Software. Englewood Cliffs, NJ: Prentice-Hall, 1988.
7. Fox, Terry L. AX.25 Amateur Packet-Radio Link-Layer Protocol (Version 2.0). Newington, CT: American Radio Relay League, 1984.
8. Friend, George E., and others. Understanding Data Communications, Indianapolis, IN: Howard W. Sams, 1984.
9. Gane, Chris, and Trish Sarson. Structured Systems Analysis: Tools and Techniques. Englewood Cliffs, NJ: Prentice-Hall, 1979.
10. Heggstad, Harold M. An Overview of Packet-Switching Communications. IEEE Communications Magazine, 22: 24-31 (April 1984).
11. Kleinrock, Leonard, and Fouad A. Tobagi. Packet Switching in Radio Channels: Part I--Carrier Sense Multiple-Access Modes and their Throughput-Delay Characteristics. IEEE Transactions on Communications, Com-23: 1400-1416 (December 1975).
12. Lerner, Barry M., and others. Issues in Packet Radio Network Design. Proceedings of the IEEE, 75: 6-20 (January 1987).

13. Martin, James, and Carma McClure. Structured Techniques for Computing. Englewood Cliffs, NJ: Prentice-Hall, 1985.
14. Mayo, Jonathan L. The Packet Radio Handbook. Blue Ridge Summit, PA: Tab Books, 1987.
15. Myers, Glenford J. The Art of Software Testing. New York: John Wiley & Sons, 1978.
16. National Communications System. Revised CCITT Recommendation X.25 1980. TIB 80-5. Washington: NCS, 11 August 1980.
17. Page-Jones, Meilir. The Practical Guide to Structured Systems Design. Boston: Yourdon Press, 1980.
18. Pressman, Roger S. Software Engineering: A Practitioner's Approach. New York: McGraw-Hill Book Company, 1987.
19. Ricci, Fred J., and Daniel Schutzer. U.S. Military Communications a C3I Force Multiplier. Rockville, MD: Computer Science Press, 1986.
20. Schildt, Herbert. Advanced C (Second Edition). Berkeley, CA: McGraw-Hill, 1988.
21. Schildt, Herbert. C Power User's Guide. Berkeley, CA: McGraw-Hill, 1988.
22. Schwartz, Mischa, and Thomas E. Stern. Routing Techniques Used in Computer Communication Networks. IEEE Transactions on Communications, COM-28: 539-552 (April 1980).
23. Tanenbaum, Andrew S. Computer Networks. Englewood Cliffs, NJ: Prentice-Hall, 1981.

Appendix A  
Advanced Electronics Applications  
Model PK-232  
Terminal Node Controller

Introduction

This appendix is broken into two sections. The first section parallels the operation of a terminal node controller with that of the familiar telephone modem. The second section presents the protocol used between interconnected terminal node controllers, AX.25.

The Advanced Electronics Applications Model PK-232 Terminal Node Controller (TNC) has found wide acceptance within the amateur radio community. The reason it was chosen for use in the hardware configuration associated with AFLC's PRN is because of its low cost and excellent reputation. However, one of the reasons for this well deserved reputation is that it serves the amateur radio community with five operational modes. Most competing amateur radio terminal node controllers do not have such scope. Only one operational mode is used in this application--the packet mode. All discussion related to the TNC concerns operation within the packet mode. One extremely beneficial feature of the PK-232 in packet mode is the ability to set up a special method for communication between the TNC and an interconnected computer. As will be detailed shortly, this special communication method, called HOST mode, allows the dialogue between the TNC and the computer to be controlled by the computer. This process, not generally available on other amateur radio terminal node controllers, was essential to the real-time processing environment of the MTS computer program.

Information for this appendix was taken from the PK-232 Operating Manual and the PK-232 Technical Manual [1,2]. Both of these publications must be obtained directly from the Advanced Electronics Applications Company.

TNC Operation and Control

In order to understand the operation of the TNC, it is beneficial to first develop an understanding of the familiar

computer modem. Three aspects of computer modem operation are developed: simple operating methods, modem control, and computer control once there is an established connection. The type of modem that will be discussed is the Hayes-type Smartmodem. After the modem is presented, similarities and differences between modem operation and control and TNC operation and control are developed.

During operation of a modem, data are received from the computer and sent out via the telephone line connected to the modem. Also, during modem operation, data received by the modem over the telephone line are sent to the computer. The reason the data coming out of the computer are not sent directly over and received from the telephone line is that the computer outputs bits which have signal shapes that do not travel efficiently over the telephone line. The modem converts the computer's output signals to signal forms that travel efficiently over the telephone line. This process requires the modem to perform a modulation and demodulation process.

During control of the modem, specifically a Hayes-type modem, commands are sent from the computer to the modem embedded in the data stream. This software method of modem control is done under manual command. Traditional communication software converts the English commands from the operator, such as connect to the modem at 257-3030, into the commands that are embedded in the data stream that flows between the computer and the modem. The embedded commands within the data stream are interpreted by the modem and processed. In this case, the modem would go off-hook, wait for a dial tone, send the telephone switch the number 257-3030, wait for a connect tone from the distant modem, respond to the distant modem connect tone, and signal the computer that the connection is made.

Once a computer-to-computer connection is established through the modems, a higher level of control is responsible for the accurate transfer of data between the computers. One method for accomplishing this responsibility is a protocol known as XMODEM [7:167], which operates by breaking a data file transfer between two computers into a number of frames. Each frame is then sent independently over the connection. The principal components of a frame are a start-of-frame character, a frame number, the data characters, an error-checking sequence, and an end-of-frame character. Upon receiving the data characters, the receiving computer checks the error-checking sequence that it generates against the error-checking sequence that was transmitted. If the

two sequences match, an acknowledgement is sent from the receiving computer to the transmitting computer. If the sequences do not match, the receiving computer sends a negative acknowledgement to the transmitting computer. Upon receipt of an acknowledgement, the transmitting computer sends the next frame. If the transmitting computer receives a negative acknowledgement, it resends the corrupted frame. This frame transfer process continues until all the fixed length frames needed to send the complete file are sent. For operation of XMODEM, both computers must be manually set to XMODEM mode, one computer established as the transmitter, the other as the receiver.

Now that the modem has been presented, the similarities and differences between the Hayes-type modem and the TNC are discussed.

Operation of the TNC is similar to the operation of the modem. At the TNC, data are received from the computer, just as with the modem; however, unlike the modem, which interconnects to another modem through use of a telephone line, the TNC interconnects to another TNC through a radio and radio channel. Data received by the TNC from the radio are processed and sent to the computer. The radio cannot receive data directly from the computer or send data directly to the computer; this is one of the functions of the TNC. In this respect, the TNC is acting as a modem, converting the computer's output signals into signaling forms that can be transmitted by the radio. Similarly, for the receive process, radio signaling forms are converted by the TNC to the computer's signaling forms. During operation, the TNC performs a modulation and demodulation process.

During control of the TNC, the process is similar to the Hayes-type modem, although there are important differences. Commands are sent from the computer to the TNC by embedding the commands in the data stream. Responses from the TNC are sent by embedding the responses in a return data stream. A special method for communicating between the computer and the TNC is possible with the TNC. This special method is referred to as HOST mode. The TNC is placed into the TNC HOST mode by commands from the computer. Once in HOST mode, the TNC only responds when requested by the computer. The specific format for this request-and-response process has two forms: command and data formats. Both are processed by sending frames which are made up of a start-of-frame character, a code character indicating whether it is a command or a data frame, the specific command or data, and

an end-of-frame character. The command frames instruct the TNC to perform some process. Examples of some processes are to connect to a specific TNC, change the station call sign, or send data received by the TNC from the radio to the computer. The data frames are used to transmit data to the TNC for transmission. These data frames would only be sent to the TNC once a connection to a distant TNC has been established. The special feature of HOST mode is that it allows the dialogue between the computer and the TNC to be controlled by the computer. Also, processing within the HOST mode can be accomplished without manual control. Like the discussion of XMODEM above, communication takes place by frames. However, unlike XMODEM, the purpose of HOST mode is to facilitate communication between the computer and the TNC, and not between two computers. No error checking is done in the transfer of information between the computer-to-TNC connection. The process that handles the TNC-to-TNC connection and handles responsibility for accurate transfer of data across that connection is the AX.25 protocol, a topic important enough to be the entire focus of the next section.

#### AX.25

During the previous presentation of the modem operation, it was mentioned that once a computer-to-computer connection is made through the modems, a higher level of control was responsible for the accurate transfer of data between the computers. A similar responsibility is given to the TNC. The TNC is responsible for ensuring that the data transferred between two TNCs are correct. This definition of correct includes a character-for-character match at the transmitting and receiving TNCs, as well as the proper ordering of those characters in a data stream. The TNC does this by using a protocol referred to as AX.25 [6]. AX.25 is the amateur radio version of the international X.25 standard [15]. The AX.25 protocol can be viewed as a sophisticated form of XMODEM. AX.25 is considered sophisticated in that it is established whenever transmission occurs between TNCs, and is invoked automatically. Like XMODEM, AX.25 uses frames to parse the message to be transmitted into small chunks, sending those frames to the destination and awaiting a positive or negative acknowledgement. The acknowledgement is based on the matching of the error-checking field at the destination TNC, similar to the XMODEM checks made at the destination computer. The principal components of an AX.25 frame are similar to an XMODEM frame and include a start-of-frame character, frame number, data characters, error-checking sequence, and end-of-frame character. One

significant difference is that the frame also has an address field. The reason for this is that the TNC transmits over a radio and radio channel. This radio channel is a broadcast medium with the possibility of reaching many TNC-radio pairs. Only the addressed TNC processes the packet. The exception to this is the principal difference between AX.25 and X.25. AX.25 allows the frame to have more than one address, with the additional addresses serving as intermediate processing nodes. If a frame has two call signs in the address field, one of those addresses is an intermediate node which will process the frame, interpret that it is not for consumption, and transmit it to the addressed destination. This powerful technique, unique to AX.25, allows for the establishing of a connection between any two nodes in a non-fully connected network, as long as a path can be made through the network, even if it involves the use of intermediate nodes.

The AX.25 process is an integral part of the TNC. This protocol serves the connected computer with the ability to relinquish responsibility for the end-to-end integrity of the data that are transmitted from the computer. This strategy relies on a quality path between the computer and its servicing TNC, since no error checking is done on this data path.

The previous discussion started with a presentation of the operation and control of the familiar computer modem. Once this foundation was set, a comparison was made between the operation and control of the modem and the TNC. Finally, the AX.25 protocol was discussed, highlighting the responsibility it has for providing end-to-end data integrity between connected TNCs.

## Appendix B

### User's Manual

#### Introduction

This User's Manual serves as a guide for AFLC's PRN MTS. The MTS is menu-driven. This guide presents the reader with the menus that are displayed on the computer monitor and an explanation of the actions that follow each menu selection.

The User's Manual is broken into three sections. The first is an example procedure to generate and transmit a message, the second is an example procedure to receive a message, and the third details each menu in the system.

All menus are presented in the same format, a vertical listing of menu items. There are limited active keyboard responses during display of a menu. These keys are the up and down arrow keys, the first letter of any menu item, the RETURN key, and CTRL-C. The up and down arrow keys move a reverse video highlight among the menu items. Only one menu item is highlighted at any time. Depressing the key corresponding to the first letter of any menu item causes the reverse video highlight to move to the selected menu item. The menu item that is highlighted in reverse video signifies the menu item selected. Once a RETURN key is hit, the selected menu item begins processing. Hitting CTRL-C causes the program to terminate and control to be returned to the operating system.

#### Transmitting a Message: An Example Procedure

This procedure gives the keystrokes necessary to generate and transmit a message. The reader is not expected to have read the detailed menu description.

1. Insert the 5 1/4-inch disk containing the computer program into disk drive B.
2. Insert the 5 1/4-inch disk containing the system routing tables in disk drive A.
3. Enter "B:STM <RETURN>".
4. The monitor will display a question about generating new routing tables. Enter "n"

5. The main menu will display on the monitor. Depress the down arrow key until the menu item GENERATE A MESSAGE is highlighted in reverse video.

6. Enter <RETURN>.

7. The monitor will display an information header describing the generate message process. Follow all directions and complete a message.

8. Upon completion of the generate message process, the main menu will be displayed. Depress the down arrow key until the menu item TRANSMIT MESSAGE is highlighted.

9. Enter <RETURN>.

10. The transmit menu will display in the monitor. Depress the down arrow key until the menu item ENTER A GENERATED MESSAGE is highlighted.

11. Enter <RETURN>.

12. The transmit submenu will display on the monitor. Depress the down arrow key until the menu item ALL MESSAGES LISTED is highlighted.

13. Enter <RETURN>. The light on disk drive A will illuminate briefly.

14. The transmit menu will display on the monitor. A sentence will appear at the bottom of the monitor, indicating that the transmit queue is not empty and that the transmit flag is disabled. Depress the down arrow key until the menu item BEGIN TO TRANSMIT is highlighted.

15. Enter <RETURN>.

16. A sentence will appear at the bottom of the monitor, explaining that there is a message being transmitted. Wait until the sentence about transmitting a message no longer appears, then depress the down arrow key until the transmit menu item RETURN TO MAIN MENU is highlighted.

17. Enter <RETURN>.

18. The main menu will display on the monitor. Depress the down arrow key until the menu item ACCESS ARCHIVED MESSAGE FILE is highlighted.

19. The access archive menu will display on the monitor. Depress the down arrow key until the menu item DISPLAY ALL TRANSMITTED MESSAGE HEADERS is highlighted.

20. Enter <RETURN>.

21. A listing of the messages that have been transmitted will display on the monitor. Confirm that the message that was just generated has been transmitted.

22. Enter <RETURN>.

23. The archive submenu will display on the monitor. Depress the down arrow key until the menu item RETURN TO MAIN MENU is highlighted.

24. The main menu will display on the monitor.

#### Receiving a Message: An Example Procedure

1. Follow the transmit procedure above to instruct a distant node to send you a message.

2. Return the system to the main menu.

3. Upon receiving a message, a sentence will appear at the bottom of the monitor, indicating that a message is being received. Wait until the sentence no longer appears, then depress the down arrow key until the menu item ACCESS ARCHIVED MESSAGE FILE is highlighted.

4. Enter <RETURN>.

5. The access archive menu will display on the monitor. Depress the down arrow key until the menu item SHOW THE RECEIVED MESSAGE HEADERS is highlighted.

6. Enter <RETURN>.

7. The list of received messages will display on the monitor.

8. Enter <RETURN>.

9. The archive submenu will display on the monitor. Depress the down arrow key until the menu item VIEW MESSAGE is highlighted.

10. Enter <RETURN>.

11. The message that was just received will be displayed completely on the screen.
12. Enter <RETURN>.
13. Depress the down arrow key until the menu item RETURN TO MAIN MENU is highlighted.
14. Enter <RETURN>.
15. The main menu will display on the monitor.

### System Menus

#### Main Menu

The main menu has four items for selection:

TRANSMIT MESSAGE  
GENERATE MESSAGE  
ACCESS ARCHIVED MESSAGE FILE  
EXIT PROGRAM

TRANSMIT MESSAGE is selected when the operator wants to transmit a message. Typically, this menu item is selected after a message has been generated and stored on a disk in drive A. Upon selection, this menu item causes the transmit menu to be displayed.

GENERATE MESSAGE is selected when the operator wants to generate a message for transmission. Upon selection of this item, an information header is displayed, outlining the message generation procedure.

ACCESS ARCHIVED MESSAGE FILE is selected when the operator wants to look at the messages stored in the system or the system's routing tables. Upon selection of this menu item, the access archive menu is displayed.

EXIT PROGRAM causes the system to stop operation, store the system's messages and routing tables on disk, reset the terminal node controller, and turn control over to the operating system. Upon selection of this menu item, there is a slight delay as the system performs exit processing before the operating system prompt appears.

Transmit Menu. The transmit menu is reached from the main menu by selecting the TRANSMIT MESSAGE menu item. The transmit menu offers the operator four menu items:

ENTER A GENERATED MESSAGE  
BEGIN TO TRANSMIT  
HOLDUP TRANSMISSION  
RETURN TO MAIN MENU

ENTER A GENERATED MESSAGE is chosen by the operator when a prepared message for transmission is on disk in drive A. Upon selection of this menu item, the transmit submenu is displayed.

BEGIN TO TRANSMIT is chosen to put the operation of the node into the active transmit mode. Upon selection of this menu item, the system's transmit flag is enabled, allowing the message at the head of the transmit queue to be transmitted.

HOLDUP TRANSMISSION is chosen to put the operation of the node into the inactive transmit mode. Upon selection of this menu item, the system's transmit flag is disabled, stopping the transmission of the next message at the head of the transmit queue.

RETURN TO MAIN MENU is chosen by the operator to display the main menu. Upon selection of this menu item, the main menu is displayed.

#### Transmit Submenu

The transmit submenu is reached from the transmit menu by selecting the ENTER A GENERATED MESSAGE menu item. The transmit submenu offers the operator four menu items:

ALL MESSAGES LISTED  
ONLY ONE MESSAGE LISTED  
ENTER TRANSMIT QUEUE LISTING  
RETURN TO MAIN MENU

ALL MESSAGES LISTED is chosen when all prepared messages on disk in drive A are to be entered. More than one message can be in the generated message file on disk in drive A. Upon selection of this menu item, all the messages in the prepared message file on disk in drive A are read into the

system and placed in the transmit queue for transmission. The transmit queue is ordered by each message's priority field.

ONLY ONE MESSAGE LISTED is chosen when only one message out of all the prepared messages on disk in drive A is to be chosen. Upon selection of this menu item, the prepared messages are read from disk and the source, destination, author, and subject are displayed. Through a reverse video highlight selection using the up and down arrow keys, the chosen message is entered into the transmit queue by depressing a RETURN key. The transmit queue is ordered by each message's priority field.

ENTER TRANSMIT QUEUE LISTING is chosen when the operator wants to enter a message directly into the transmit queue. Upon selection of this menu item, message field prompts appear and the cursor moves among the fields as the operator enters information. Once all the fields have been entered, the message is stored in the transmit queue. The transmit queue is ordered by each message's priority field.

RETURN TO MAIN MENU is chosen by the operator to display the main menu. Upon selection of this menu item, the main menu is displayed.

Generate Message. The generate message procedure is reached from the main menu by selection of the GENERATE A MESSAGE menu item. The generate message process does not display a menu; instead, it begins by displaying the following information header:

#### GENERATE MESSAGE PROGRAM

This program will ask questions about the message, such as the message's source node, destination node, message text, etc.

Your answers are stored in a disk file named a:messages.

Type any key to begin.

To quit, immediately type a return after the first question.

Once the operator types a key, the message fields are displayed. The cursor then moves to the starting field and waits for the operator's entry. The monitor echoes the keyboard entries, placing them in the current message field being worked. When all the message fields have been entered, the message is stored in RAM and another blank message template is displayed. Should the operator want to enter another message, the message field entry procedure detailed above is followed. When no further messages are to be generated, the RETURN key is depressed at the first message field entry point, causing all the generated messages to be stored on disk.

Access Archive Menu. The access archive menu is reached from the main menu by selecting the ACCESS ARCHIVED MESSAGE FILE menu item. The access archive menu offers the operator seven menu items:

- DISPLAY ALL TRANSMITTED MESSAGE HEADERS
- SHOW THE RECEIVED MESSAGE HEADERS
- THE TRANSMIT QUEUE
- A SPECIFIC NODE'S TRANSMIT AND RECEIVE MESSAGE HEADERS
- MESSAGE HEADERS FOR A SPECIFIC TIME PERIOD
- CURRENT ROUTING TABLE
- RETURN TO MAIN MENU

DISPLAY ALL TRANSMITTED MESSAGE HEADERS is chosen when the operator wants to see a listing of all messages that have been transmitted. Upon selection of this menu item, the monitor displays a listing of all the messages transmitted.

SHOW THE RECEIVED MESSAGE HEADERS is chosen when the operator wants to see a listing of all the messages that have been received. Upon selection of this menu item, the monitor displays a listing of all the messages received.

THE TRANSMIT QUEUE is chosen when the operator wants to see a listing of messages in the transmit queue. Upon selection of this menu item, the monitor displays a listing of the messages in the transmit queue.

A SPECIFIC NODE'S TRANSMIT AND RECEIVE MESSAGE HEADERS is chosen when the operator wants to see a listing of messages transmitted from the local node to the specified node and the messages received by the local node from the specified node. Upon selection of this menu item, a prompt for

the node to view is followed by a listing of the specified node's transmitted and received messages.

MESSAGE HEADERS FOR A SPECIFIC TIME PERIOD is chosen when the operator wants to see a listing of transmitted and received message headers that have been processed during a specific time period. Upon selection of this menu item, a prompt for the time period is followed by a listing of messages processed within the specified time period.

Upon selection, these first five menu items have similar responses. The monitor displays a listing of messages: the top entry is highlighted in reverse video, the down arrow key moves the highlighted message down the listing, and depressing the RETURN key causes an archive submenu to pull down over the listing.

CURRENT ROUTING TABLE is chosen when the operator wants to see the current routing tables. Upon selection of this menu item, the current routing tables are listed. A reverse video highlight, under the down arrow control, can be moved down the routing table listing. When the routing table to be operated on is highlighted, the RETURN key is depressed and the routing table submenu pulls down over the routing table listing.

RETURN TO MAIN MENU is chosen when the operator wants to return to the main menu. Upon selection of this menu item, the main menu is displayed.

#### Archive Submenu

The archive submenu is reached upon depressing the RETURN key from within any of the message listings associated with the first five access archive menu items. The archive submenu offers the operator seven menu items:

- PRINT MESSAGE
- STORE MESSAGE AS
- VIEW MESSAGE
- DELETE MESSAGE
- LINKED LIST PRINT
- GO TO ARCHIVE MENU
- RETURN TO MAIN MENU

PRINT MESSAGE is chosen when the operator wants to print the message that is highlighted in reverse video. Upon selection of this menu item, the message that is highlighted

in reverse video is printed on the system printer. Following printing, the listing that was displayed prior to entering the archive submenu is displayed. The archive submenu is not displayed.

STORE MESSAGE AS is chosen when the operator wants to store the message highlighted in reverse video on disk. Upon selection, a prompt requesting the file store name is followed by a brief illumination of the disk drive's status light. Following this, the listing that was displayed prior to entering the archive submenu is displayed. The archive submenu is not displayed.

VIEW MESSAGE is chosen when the operator wants to see the entire message on the monitor. During display of the message listing, only a limited number of each message's fields is shown. Upon selection of this menu item, the message that is highlighted in reverse video is shown complete on the monitor. When the operator has finished viewing the message, the RETURN key is depressed. The monitor then displays the listing that was displayed prior to entering the archive submenu. The archive submenu is not displayed.

DELETE MESSAGE is chosen when the operator wants to delete the message that is highlighted in reverse video. Upon selection of this menu item, the message that is highlighted is deleted. The monitor then displays the abbreviated version of the listing that was displayed prior to entering the archive submenu. The archive submenu is not displayed.

LINKED LIST PRINT is chosen when the operator wants the entire listing printed. Upon selection of this menu item, the message listing that appears on the monitor is sent to the printer. The monitor then displays the listing that was displayed prior to entering the archive submenu. The archive submenu is not displayed.

GO TO ARCHIVE MENU is chosen when the operator wants to go back to the access archive menu. Upon selection of this menu item, the monitor is first cleared and then the access archive menu appears.

RETURN TO MAIN MENU is chosen when the operator wants to return to the main menu. Upon selection of this menu item, the main menu is displayed.

### Routing Table Submenu

The routing table submenu is reached by depressing the RETURN key from within the routing table listing associated with the access archive menu. The routing table submenu offers the operator seven menu items:

- ADD A NEW ROUTING TABLE ENTRY
- MODIFY AN EXISTING ROUTING TABLE ENTRY
- VIEW ENTIRE ENTRY
- DELETE THE HIGHLIGHTED ROUTING TABLE ENTRY
- ENTER NEW ROUTING TABLE FROM DISK
- STORE CURRENT ROUTING TABLE ON DISK
- RETURN TO MAIN MENU

ADD A NEW ROUTING TABLE ENTRY is chosen when the operator wants to add a new destination to the routing table. Upon selection of this menu item, the monitor is cleared and the routing table fields are displayed. When the fields have been filled by the operator, the monitor displays the routing table listing including the newest entry.

MODIFY AN EXISTING ROUTING TABLE ENTRY is chosen when the operator wants to modify the highlighted routing table entry. Upon selection of this menu item, the monitor is cleared and then displays the entire routing table entry. The cursor is in the first routing table field. The operator can keep the current routing table field entry by depressing the RETURN key. To change the routing table field entry, the operator types in the new information at the appropriate field. When all fields have been worked, the monitor is cleared, and then displays all routing table listings.

VIEW ENTIRE ENTRY is chosen when the operator wants to view the complete routing table entry that is highlighted. When the routing table listing is displayed, the complete routing table entry is not shown. Upon selection of this menu item, the routing table entry that is highlighted in reverse video is displayed on the monitor. When the viewing is finished, the operator depresses the RETURN key. The monitor then displays the routing table listing.

DELETE THE HIGHLIGHTED ROUTING TABLE ENTRY is chosen when the operator wants to delete the highlighted routing table entry. Upon selecting this menu item, the highlighted menu item is deleted, and the abbreviated routing table listing is displayed.

ENTER NEW ROUTING TABLE FROM DISK is chosen when the operator wants to have the current routing table listing replaced with the routing table listing on disk. Upon selection of this menu item, a brief illumination of the disk drive light is followed by a display on the monitor of the new routing table listing read from disk.

STORE CURRENT ROUTING TABLE ON DISK is chosen when the operator wants to store the current routing table on disk. Upon selection of this menu item, a brief illumination of the disk drive light is followed by a display on the monitor of the current routing table listing.

RETURN TO MAIN MENU is chosen when the operator wants to return to the main menu. Upon selection of this menu item, the main menu item is displayed.

Appendix C

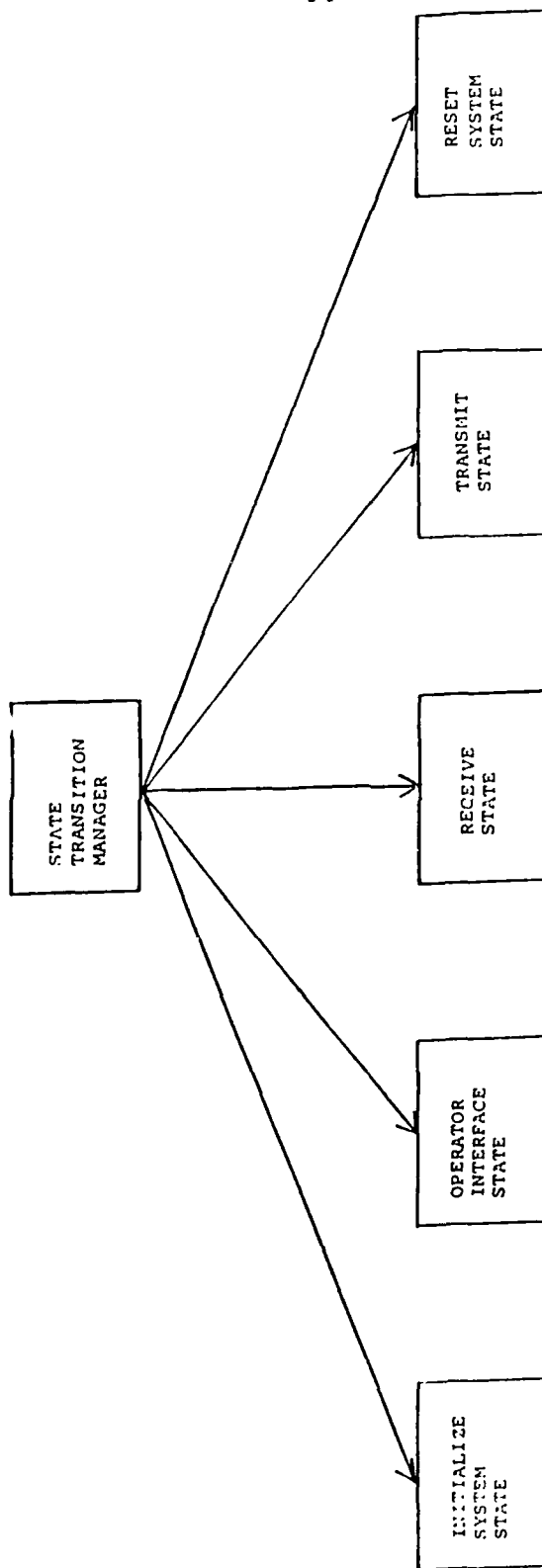


Figure C-1 System Structure Chart

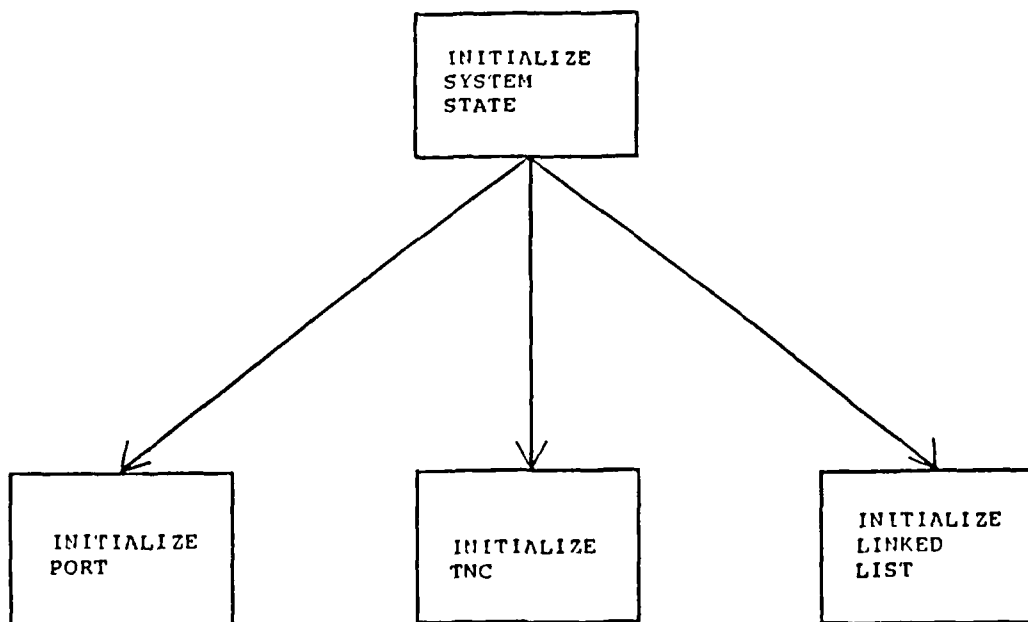


Figure C-2 Initialize System State Module Structure Chart

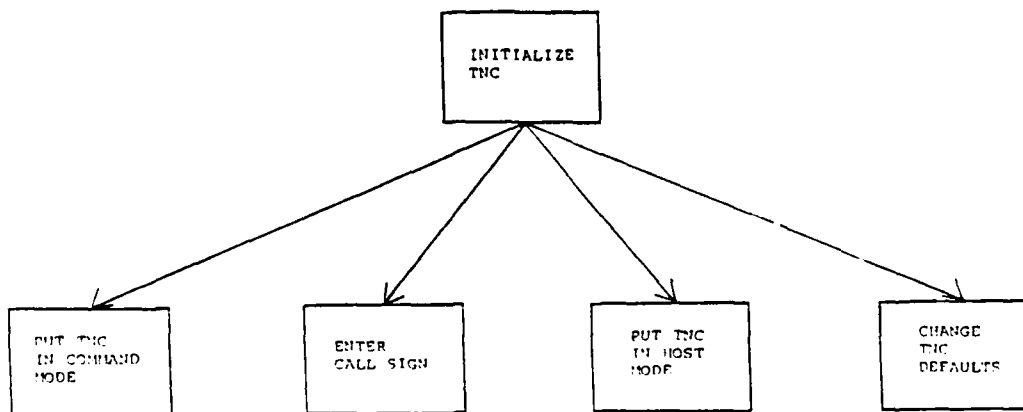


Figure C-3 Initialize TNC Module Structure Chart

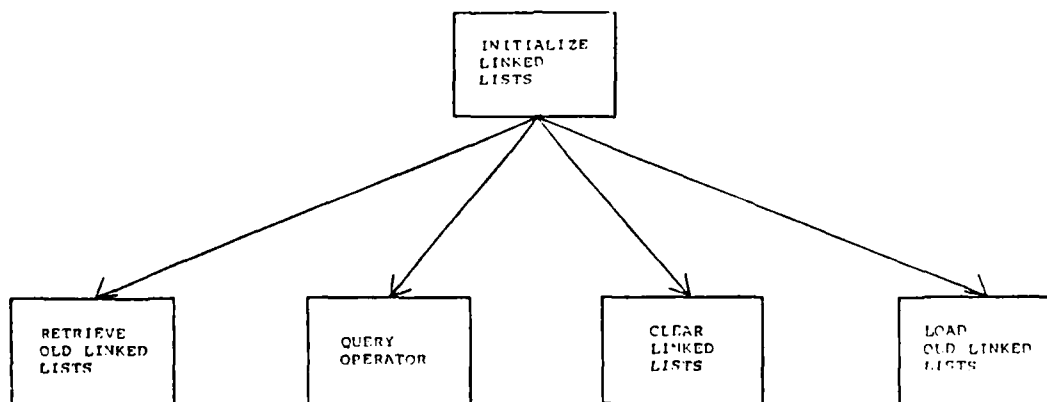


Figure C-4 Initialize Linked Lists Module Structure Chart

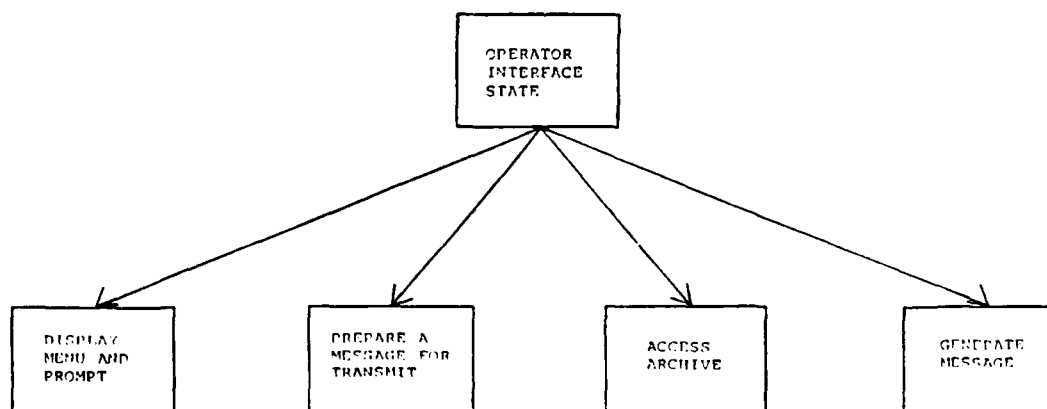


Figure C-5 Operator Interface State Module Structure Chart

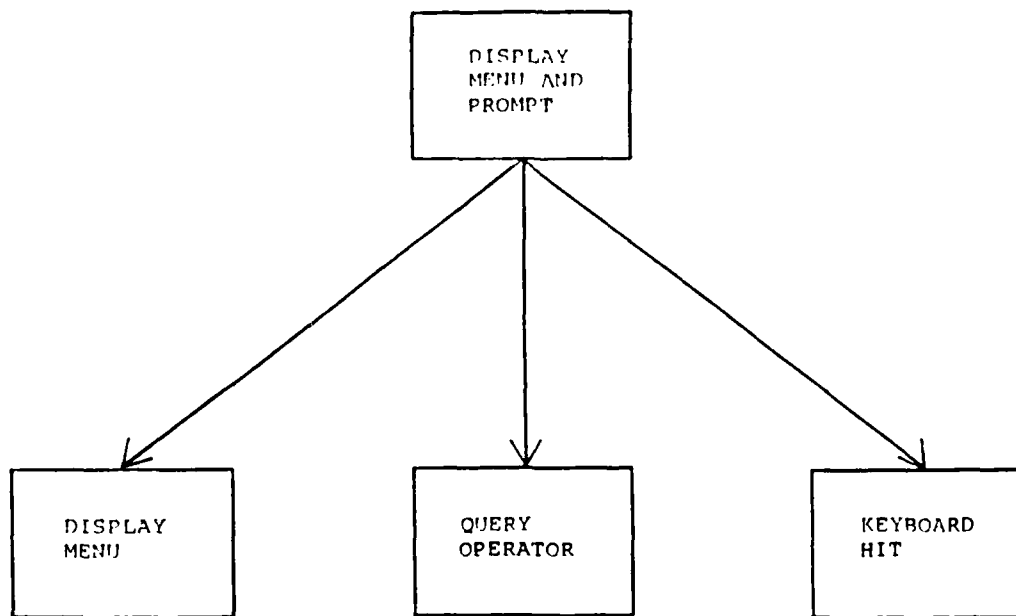


Figure C-6 Display Menu and Prompt Module Structure Chart

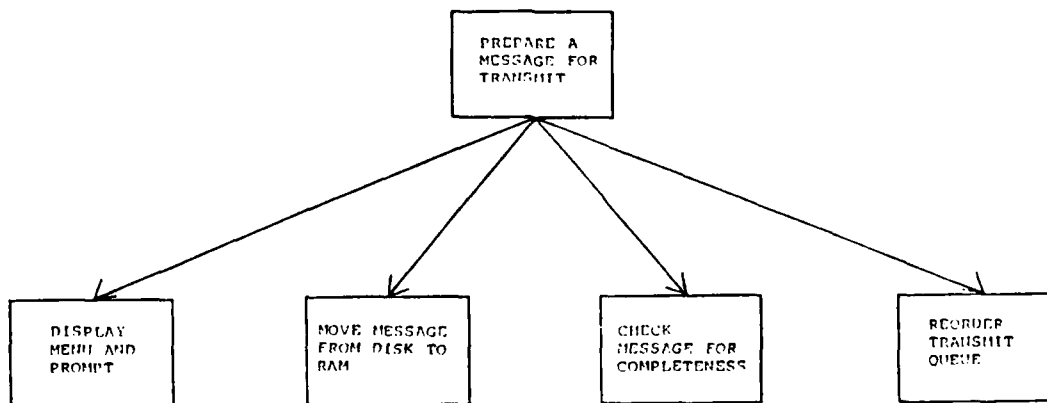


Figure C-7 Prepare Message for Transmit Module Structure Chart

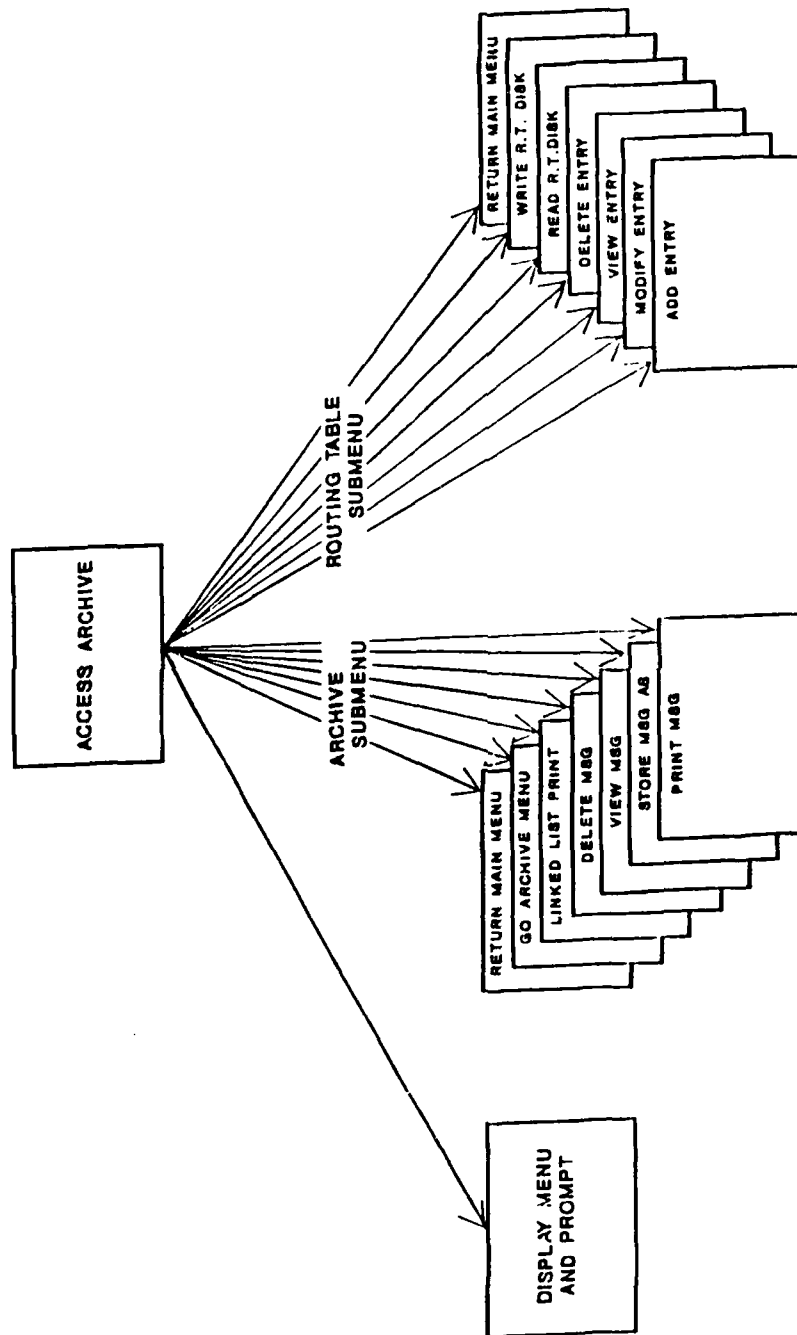


Figure C-8 Access Archive Module Structure Chart

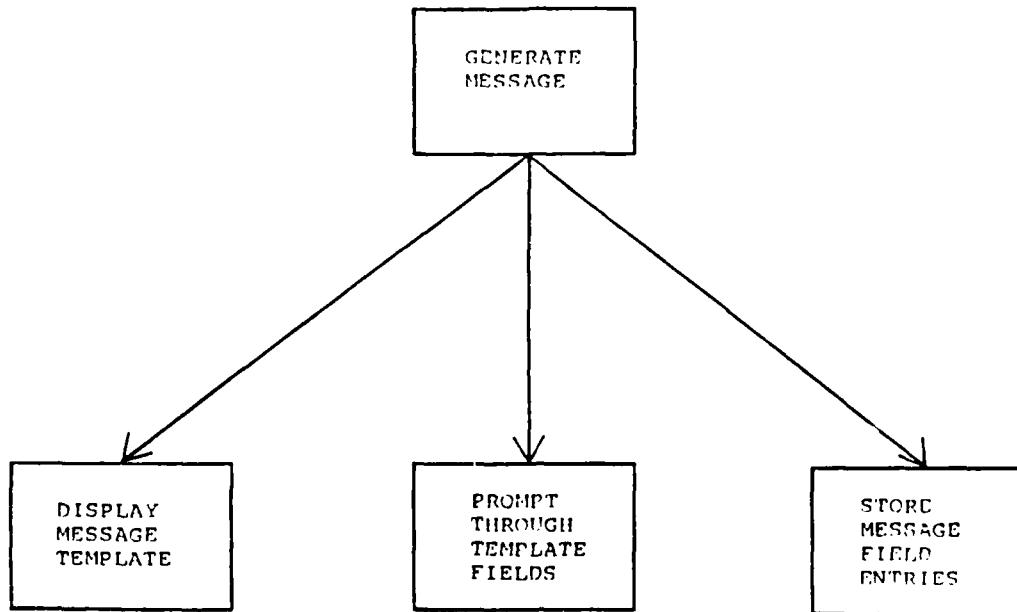


Figure C-9 Generate Message Module Structure Chart

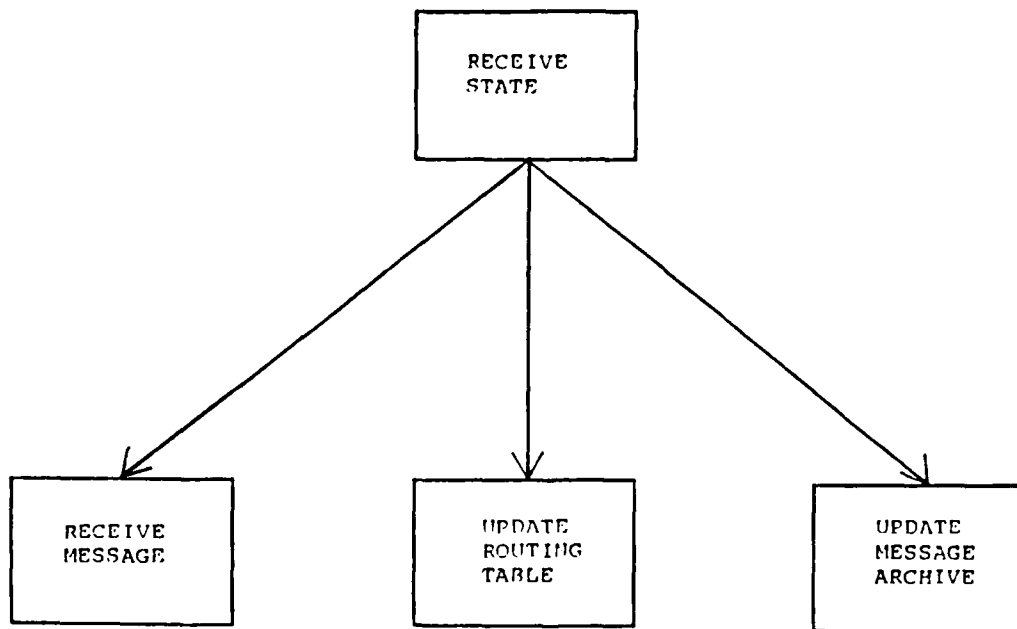


Figure C-10 Receive State Module Structure Chart

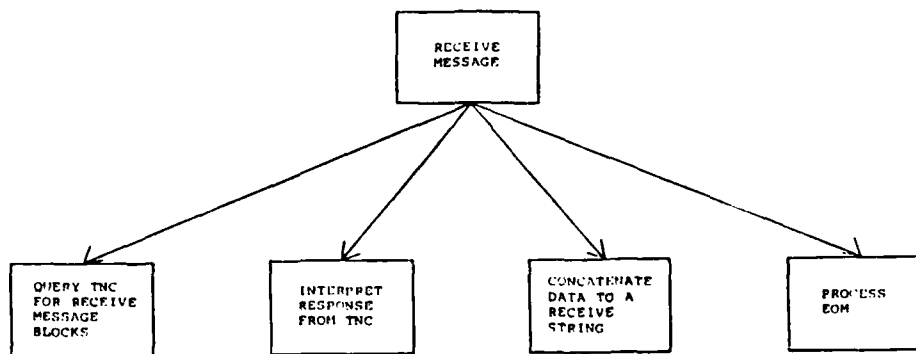


Figure C-11 Receive Message Module Structure Chart

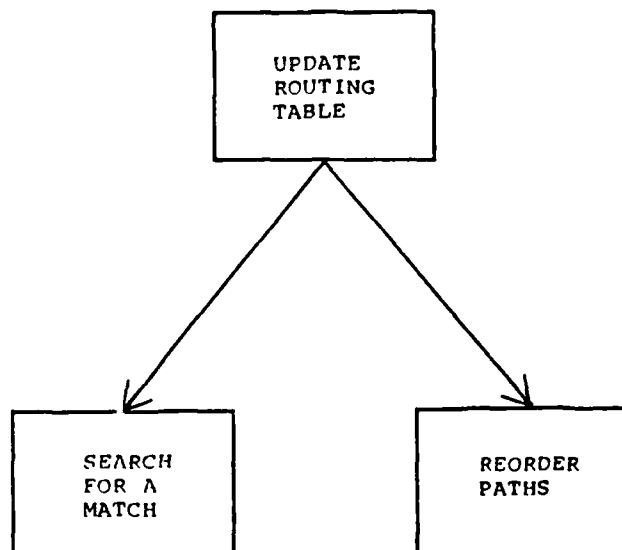


Figure C-12 Update Routing Table (Receive) Module Structure Chart

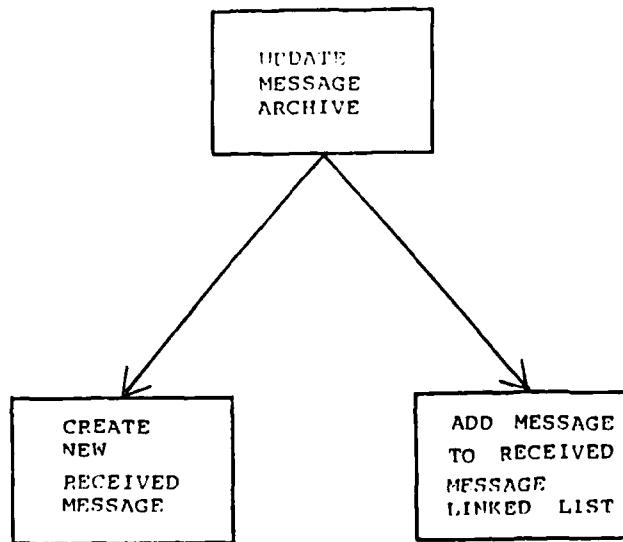


Figure C-13 Update Received Message Archive Module Structure Chart

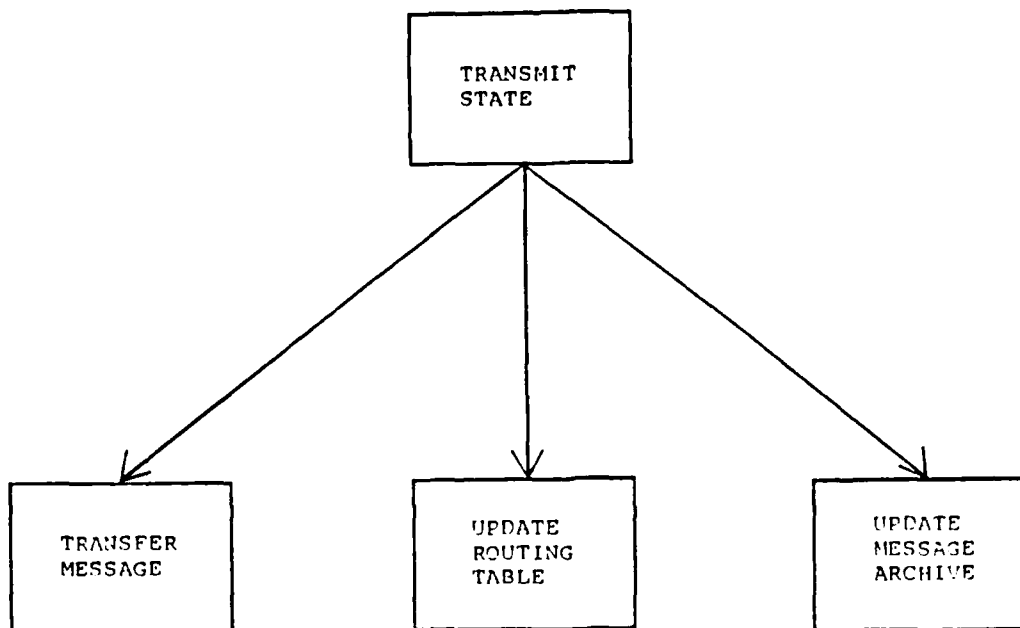


Figure C-14 Transmit State Module Structure Chart

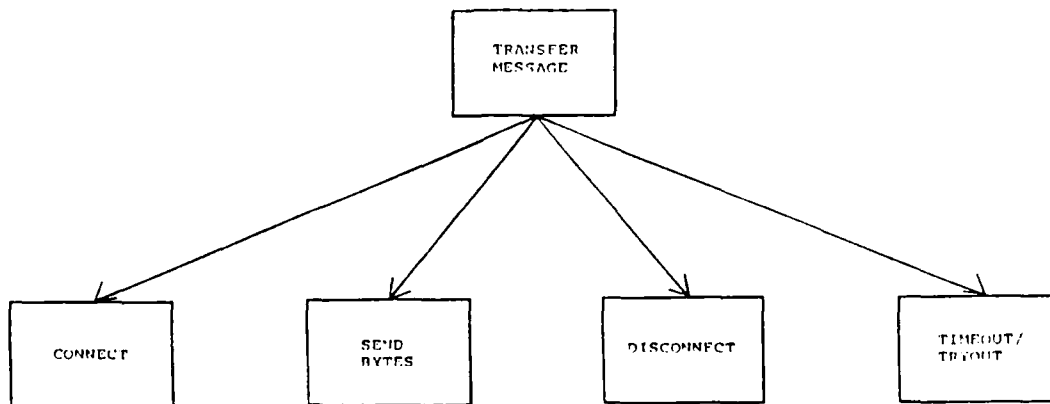


Figure C-15 Transfer Message Module Structure Chart

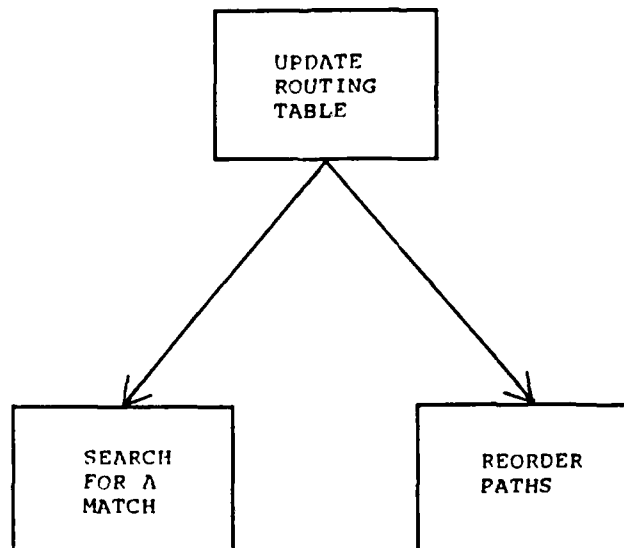


Figure C-16 Update Routing Table (Transmit) Module Structure Chart

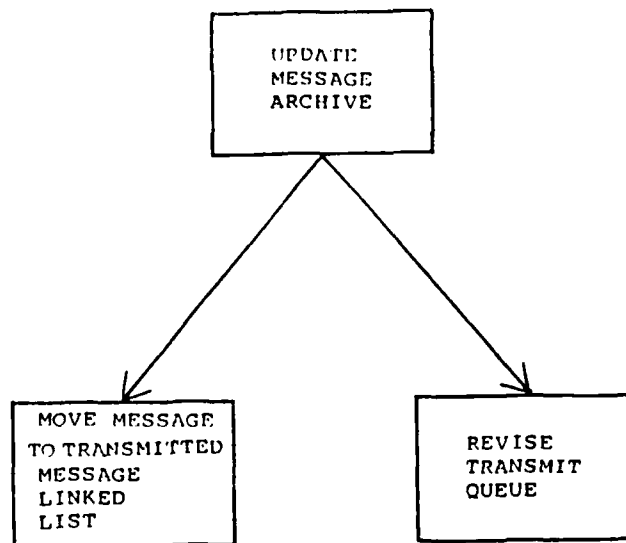


Figure C-17 Update Transmitted Message Archive Module Structure Chart

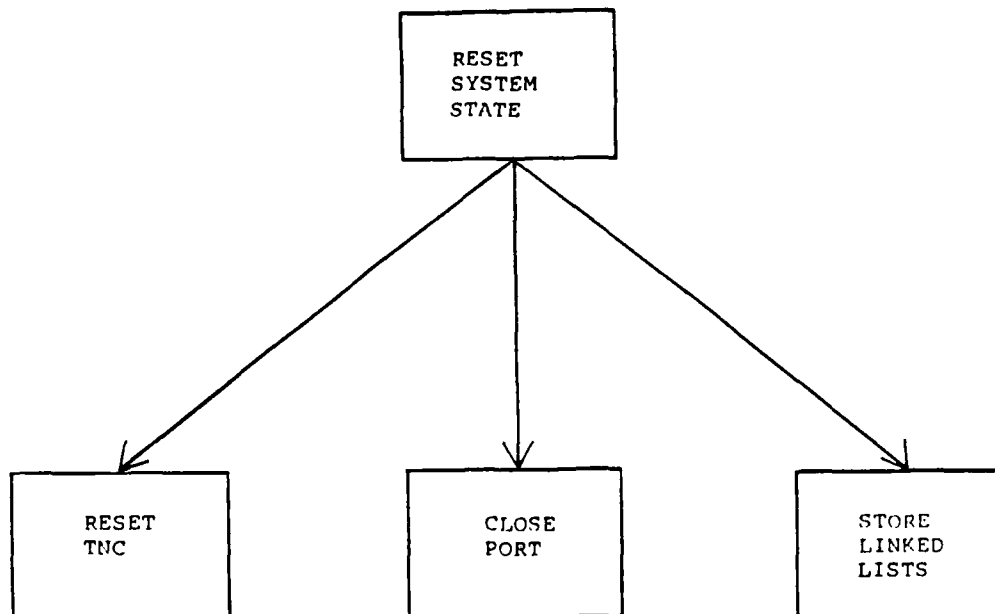


Figure C-18 Reset System State Module Structure Chart

# Appendix D. DATA DICTIONARY

Page

## DATA PROCESSES

CHECK FOR COMPLETENESS . . . . .	D.3
DETERMINE ROUTE . . . . .	D.3
DSP MSG TEMPLATE & QUERY . . . . .	D.4
INTERFACE TNC . . . . .	D.4
NOTIFY OPERATOR . . . . .	D.5
PARSE IN MSG . . . . .	D.6
PARSE OUT MSG . . . . .	D.6
RCONNECT/DISCONNECT . . . . .	D.7
RCV STATISTICS . . . . .	D.7
RQUEUE . . . . .	D.7
TCONNECT/DISCONNECT . . . . .	D.8
TQUEUE . . . . .	D.8
TRANSFER MSG . . . . .	D.8
TX STATISTICS . . . . .	D.9
UPDATE ROUTING ALGORITHM . . . . .	D.9

## DATA STRUCTURES

CONN . . . . .	D.10
DESTINATION . . . . .	D.10
EOM POINTER . . . . .	D.10
GEN STRUCTURE . . . . .	D.10
INPUT . . . . .	D.10
MSG . . . . .	D.10
MSG POINTER . . . . .	D.10
NEXT RMSG POINTER . . . . .	D.11
NEXT TMSG POINTER . . . . .	D.11
ORDERED R POINTERS . . . . .	D.11
ORDERED T POINTERS . . . . .	D.11
RCV C/D . . . . .	D.11
RMSG POINTER . . . . .	D.11
RMSG PKT . . . . .	D.11
ROUTE . . . . .	D.11
RROUTE . . . . .	D.11
SERIAL DATA . . . . .	D.12
STATUS . . . . .	D.12
TEMPLATE . . . . .	D.12
TMSG PKT . . . . .	D.12

# Appendix D: DATA DICTIONARY (Cont'd)

	Page
TMSG POINTER . . . . .	D.12
TMSG PRIORITY HEADER . . . . .	D.12
TROUTE . . . . .	D.12
TX C/D . . . . .	D.13
VALIDATION VALUES . . . . .	D.13
<u>DATA STORES</u>	
CUSTOMER'S MESSAGE . . . . .	D.14
MSG STORE . . . . .	D.14
RMSG ARCHIVE . . . . .	D.14
RMSG POINTERS . . . . .	D.14
RMSG QUEUE . . . . .	D.15
ROUTING FILE . . . . .	D.15
TMST QUEUE . . . . .	D.15
VALID PARAMETERS . . . . .	D.15

## DATA PROCESSES

PROCESS: CHECK FOR COMPLETENESS  
DESCRIPTION: Review a message for completeness. Each message entered must pass through this review process. Once through, the message becomes valid and goes on to be entered in the transmit message queue. Append the operator's initials to the message.

INPUTS: SEND  
MSG  
VALID PARAMETERS

LOGIC SUMMARY: Upon sensing a SEND from the active operator interface menu, compare each of the 10 message elements with its associated valid parameter. If all message elements match VALID PARAMETERS, pass. Provide an indication on the operator's screen indicating the message passed, STATUS. If any element cannot match to an associated valid parameter, provide a failed indication on the operator's screen, along with an error pointer to the failed message element, STATUS. Transfer the complete message, appending operator's initials, to TMSG QUEUE. Provide the TQUEUE process with the location of the newly completed message, TMSG POINTER.

OUTPUTS: TMSG POINTER  
TMSG  
STATUS

PHYSICAL REF: TX

PROCESS: DETERMINE ROUTE  
DESCRIPTION: The heart of the routing algorithm. Given a destination a "best" route is determined.

INPUTS: SOURCE  
DESTINATION  
ROUTE

LOGIC SUMMARY: Given DESTINATION determine if the direct path can be used between the source and destination. If source to destination is on file as being "recently" successful, use, TROUTE. If unsuccessful in establishing a connection on the direct route, given back DESTINATION, then determine a "best" path using an intermediate node or intermediate nodes, TROUTE. If connection is successful, record as a "good" path, ROUTE. If unsuccessful determine another path using a

different sequence of intermediate nodes, TROUTE. Continue process until TIMEOUT or TRYOUT is reached. All connect paths are passed to TCONNECT/DISCONNECT.

OUTPUTS: TROUTE  
ROUTE

PHYSICAL REF: TX

PROCESS: DSP MSG TEMPLATE & QUERY

DESCRIPTION: During message generation, a template of a blank message is displayed on the customer's computer screen. The cursor prompts the customer for data. The customer's keyboard responses are stored in RAM.

INPUTS: GEN MSG  
INPUT

LOGIC SUMMARY: Upon receiving a GEN MSG command, display on the customer's computer screen the message template, TEMPLATE, and query for INPUT. As INPUT arrives, store in RAM and advance cursor. Provide TRANSFER MSG process with MSG POINTER after entire message has been input.

OUTPUTS: TEMPLATE  
MSG POINTER  
INPUT

PHYSICAL REF: GENERATE

PROCESS: INTERFACE TNC

DESCRIPTION: A serial data stream is transferred to, and received from, the TNC via the computer's serial port. The serial port is directly connected to the computer's Universal Asynchronous Receiver Transmitter (UART). Access through the computer's UART is controlled by the INTERFACE TNC process. The process transfers the data coming from the computer program's processes into the computer's UART. Also, data being received at the UART from the TNC is decoded to determine which process should receive the data.

INPUT: SERIAL DATA  
TX C/D  
RCV C/D  
TMSG PKT  
RMSG PKT

LOGIC SUMMARY: During establishment of a connection during a transmit, transfer a connect path, TX C/D, to the UART transmit buffer. Query and report back the TNC's success at establishing the connection to the TCONNECT/DISCONNECT process. As the data stream, TMSG PKT, associated with the transmit, begins to arrive from the processm PARSE MSG OUT, transfer to the UART transmit buffer. Query and report back the TNCs being ready for more packets. Continue to transfer TMSG PKTs in this manner until a TX C/D signals a disconnect. Transfer this request to the UART transmit buffer. Query and report the TNC's response back to TCONNECT/DISCONNECT.

During establishment of a connection on a receive, signal the RCONNECT/DISCONNECT process, RCV C/D, of the incoming path. Continue to monitor the UART receiver buffer, transferring arrivals, RMSG PKTs, to the PARSE IN MSG process. Monitor all UART receiver buffer arrivals for a disconnect command. When a disconnect arrives, signal the RCONNECT/DISCONNECT process through an RCV C/D that a disconnect has been processed by the TNC.

OUTPUT: Continuously query the TNC for incoming connect requests.

SERIAL DATA  
RCV C/D  
TX C/D

PHYSICAL REF: TX  
RCV

PROCESS: NOTIFY OPERATOR

DESCRIPTION: Through visual and audible signals, inform the operator of a received message. Provide an output of the message to the printer on command from the operator.

INPUT: NEXT RMSG POINTER  
RMSG

LOGIC SUMMARY: Check contents of RMSG POINTERS. As long as RMSG POINTERS is not empty, process a received message. Identify next message location, NEXT MSG POINTER. Indicate to

operator via the computer screen visually, and the computer bell audibly, when a message has been received, RRMSG. At the operator's option, provide the received message to the computer screen, printer, or both.

OUTPUT: RMSG  
PHYSICAL REF: RCV

PROCESS: PARSE IN MSG  
DESCRIPTION: Message is sent in packets. These packets must be taken in by the receiver and processed to form a whole message. Packets are transferred as they are received into RAM storage. When the entire message has been received, an indication is provided to the follow-on process.

INPUTS: CONN  
RMSG PKT

LOGIC SUMMARY: Upon receiving a CONN signal, the process implements a RMSG POINTER and begins to receive RMSG PKTS. These packets are each sent to the RAM location specified by RMSG POINTER. Once all packets are received, the RMSG POINTER is sent back to the RCONNECT/DISCONNECT process.

OUTPUTS: RMSG PKT  
RMSG POINTER  
PHYSICAL REF: RCV

PROCESS: PARSE OUT MSG  
DESCRIPTION: Message is broken into packets and sent to TNC. TNC can only handle one packet at a time. Process feeds the TNC message packets upon demand until the entire message is transmitted.

INPUT: TMSG POINTER  
TMSG PKT

LOGIC SUMMARY: When TMSG POINTER arrives, begin to pull from TMSG QUEUE at location specified by TMSG POINTER. Segment message into packets, TMSG PKT, and forward to INTERFACE TNC process for transfer to TNC. Upon reaching the end of the message, send EOM POINTER to TCONNECT/DISCONNECT process and TX STATISTICS process.

OUTPUT: TMSG PKT  
EOM POINTER  
PHYSICAL REF: TX

PROCESS: RCONNECT/DISCONNECT  
 DESCRIPTION: Process sets in motion receipt of a message. Also involved in last step of transmission of a message, disconnecting.  
 INPUT: RCV C/D  
 RMSG POINTER  
 LOGIC SUMMARY: Upon receipt of a connect path, RCV C/D, from INTERFACE TNC, process pass CONN to PARSE IN MSG. Pass SOURCE to UPDATE ROUTING ALGORITHM. Upon a disconnect request, RCV C/D, transfer RMSG POINTER received from PARSE IN MSG process to RQUEUE process.  
 OUTPUT: SOURCE  
 RMSG POINTER  
 CONN  
 PHYSICAL REF: RCV

PROCESS: RCV STATISTICS  
 DESCRIPTION: Once a message is received, it will be archived on a 5 1/4-inch disk. This archiving process records time received. The message plus appended time is transferred onto a 5 1/4-inch disk.  
 INPUT: RMSG POINTER  
 RMSG  
 LOGIC SUMMARY: When an RMSG POINTER is received, the indicated message is transferred to a 5 1/4-inch disk from RAM. A time stamp is appended to the message in message field TIME RECEIVED.  
 OUTPUT: FMSG PLUS TIMESTAMP PLUS...  
 PHYSICAL REF: RCV

PROCESS: RQUEUE  
 DESCRIPTION: Upon indication of a newly received message, the process reorders the RMSG POINTERS such that the highest priority message is top pointer.  
 INPUTS: RMSG POINTER  
 RMSG PRIORITY HEADER  
 RMSG POINTERS (DATA STORE)  
 LOGIC SUMMARY: Receive RMSG POINTER. Read RMSG POINTERS. Read new message's priority. Compare. Reorder. Write back into RMSG POINTERS.  
 OUTPUTS: ORDERED R POINTERS  
 PHYSICAL REF: RCV

PROCESS: TCONNECT/DISCONNECT  
 DESCRIPTION: Connect paths establish connection through the TNC with the destination node. Connect paths can have intermediate nodes in connect path. Disconnect path follows connect path, serving to complete transmission.  
 INPUT: NEXT TMSG POINTER  
 TMSG DESTINATION  
 TROUTE  
 EOM POINTER  
 LOGIC SUMMARY: Whenever TMSG POINTERS is not empty, determine the msg destination through the network routine algorithm. Routing algorithm returns a connect path that might be successful. If connection using this path is not successful, repeat the routing algorithm step. When connection is successful, begin the transfer of the message process. Once message has been transferred, disconnect.  
 OUTPUT: TX C/D  
 TMSG POINTER  
 DESTINATION  
 PHYSICAL REF: TX

PROCESS: TQUEUE  
 DESCRIPTION: Scans newly arrived message's priority and compares that priority with other transmit queued messages. Orders the transmit message pointers so that highest priority message will be transmitted before a lower priority message.  
 INPUTS: TMSG POINTER  
 TMSG PRIORITY HEADER  
 TMSG POINTERS (DATA STORE)  
 LOGIC SUMMARY: Receive TMSG POINTER. Read TMSG POINTERS. Compare old pointers' priorities with new message priority. Reorder pointers' highest to lowest priority. Store in TMSG POINTERS.  
 OUTPUTS: ORDERED T POINTERS  
 PHYSICAL REF: TX

PROCESS: TRANSFER MSG  
 DESCRIPTION: During message generation, after customer has input all data, transfer message from RAM to a 5 1/4-inch disk.  
 INPUT: MSG POINTER  
 MSG  
 LOGIC SUMMARY: Upon receipt of a MSG POINTER, the MSG is transferred from RAM to a 5 1/4-inch disk.  
 OUTPUT: MSG  
 PHYSICAL REF: GENERATE

PROCESS: TX STATISTICS  
 DESCRIPTION: Once a message has been transmitted, it will be archived on a 5 1/4-inch disk. This archiving process will record the time the message was transmitted.  
 INPUT: TMSG  
 TMSG POINTER  
 EOM POINTER  
 LOGIC SUMMARY: When TMSG/EOM POINTER is transferred, take a timestamp and append to message addresses by TMSG POINTER. Transfer entire message to TMSG ARCHIVE on a 5 1/4-inch disk.  
 OUTPUT: TMSG PLUS TIMESTAMP PLUS...  
 PHYSICAL REF: TX

PROCESS: UPDATE ROUTING ALGORITHM  
 DESCRIPTION: When a received message comes in, the path that message took will be used to update routing tables to indicate a "best" path available.  
 INPUTS: SOURCE  
 ROUTE  
 LOGIC SUMMARY: Upon a receive connection being established, the path the incoming message takes is considered to be the "best" current path through the network. The received path will be interpreted and recorded as the "best" path for a message connection having that destination. The "best" status will last until a different message comes in from the same source or REDO time is reached.  
 OUTPUTS: ROUTE  
 PHYSICAL REF: RCV

## DATA STRUCTURES

STRUCTURE:	CONN
DFD:	RCV
PATH:	RCONNECT/DISCONNECT, PARSE IN MSG
COMPONENTS:	CONN
STRUCTURE:	DESTINATION
DFD:	TX
PATH:	TCONNECT/DISCONNECT
COMPONENTS:	DESTINATION
STRUCTURE:	EOM POINTER
DFD:	TX
PATH:	PARSE OUT MSG, TCONNECT/DISCONNECT
COMPONENTS:	EOM POINTER
STRUCTURE:	GEN STRUCTURE
DFD:	GENERATE
PATH:	CUSTOMER, DSP MSG TEMPLATE & QUERY
COMPONENTS:	GEN MSG
STRUCTURE:	INPUT
DFD:	GENERATE
PATH:	KEYBOARD, DSP MSG TEMPLATE & QUERY
COMPONENTS:	DSP MSG TEMPLATE & QUERY, MSG STORE (RAM)
	INPUT
STRUCTURE:	MSG
DFD:	GENERATE
	TX
PATH:	MSG STORE (RAM), TRANSFER MSG
	TRANSFER MSG, CUSTOMER'S MSG (FLOPPY)
	CUSTOMER'S MSG, CHECK FOR COMPLETENESS
COMPONENTS:	SOURCE
	DESTINATION
	DATE
	SUBJECT
	AUTHOR
	PRIORITY
	SECURITY CLASSIFICATION
	MSG TEXT
STRUCTURE:	MSG POINTER
DFD:	GENERATE
PATH:	DSP MSG TEMPLATE & QUERY, TRANSFER MSG
COMPONENTS:	MSG POINTER

STRUCTURE:	NEXT RMSG POINTER
DFD:	RCV
PATH:	RMSG POINTERS, NOTIFY OPERATOR
COMPONENTS:	NEXT MSG POINTER
STRUCTURE:	NEXT TMSG POINTER
DFD:	TX
PATH:	TMSG POINTERS, TCONNECT/DISCONNECT
COMPONENTS:	NEXT TMSG POINTER
STRUCTURE:	ORDERED R POINTERS
DFD:	RCV
PATH:	RQUEUE, RMSG POINTERS
COMPONENTS:	ORDERED R POINTERS
STRUCTURE:	ORDERED T POINTERS
DFD:	TX
PATH:	TQUEUE, TMSG POINTERS
COMPONENTS:	ORDERED T POINTERS
STRUCTURE:	RCV C/D
DFD:	RCV
PATH:	INTERFACE TNC, RCONNECT/DISCONNECT
COMPONENTS:	HEADER CONNECT PATH DISCONNECT PATH
STRUCTURE:	RMSG POINTER
DFD:	RCV
PATH:	RCONNECT/DISCONNECT, RQUEUE
COMPONENTS:	RMSG POINTER
STRUCTURE:	RMSG PKT
DFD:	RCV
PATH:	INTERFACE TNC, PARSE IN MSG PARSE IN MSG, RMSG QUEUE (RAM)
COMPONENTS:	RMSG PKT
STRUCTURE:	ROUTE
DFD:	TX
PATH:	DETERMINE ROUTE, ROUTING FILE
COMPONENTS:	DESTINATION INTERMEDIATE NODE 1 INTERMEDIATE NODE 2 INTERMEDIATE NODE 3 INTERMEDIATE NODE 4
STRUCTURE:	SEND
DFD:	TX
PATH:	OPERATOR, CHECK FOR COMPLETENESS
COMPONENTS:	MENU SEND FILENAME

STRUCTURE:	SERIAL DATA
DFD:	TX
	RCV
PATH:	INTERFACE TNC, TERMINAL NODE CONTROLLER
	TERMINAL NODE CONTROLLER, INTERFACE TNC
COMPONENTS:	SERIAL DATA
STRUCTURE:	STATUS
DFD:	TX
PATH:	CHECK FOR COMPLETENESS, OPERATOR
COMPONENTS:	FILE NAME
	ERROR POINTER
STRUCTURE:	TEMPLATE
DFD:	GENERATE
PATH:	DSP MSG TEMPLATE & QUERY, CRT
COMPONENTS:	A BOARDER
	"DESTINATION:"
	"DATE/TIME:"
	"SUBJECT:"
	"AUTHOR:"
	"PRIORITY:"
	"SECURITY CLASSIFICATION:"
	"MSG:"
STRUCTURE:	TMSG PKT
DFD:	TX
PATH:	TMSG PKTS, PARSE OUT MSG
COMPONENTS:	PARSE OUT MSG, INTERFACE TNC
	TMSG PKT
STRUCTURE:	TMSG POINTER
DFD:	TX
PATH:	CHECK FOR COMPLETENESS, TQUEUE
COMPONENTS:	TMSG POINTER
	PRIORITY
STRUCTURE:	TMSG PRIORITY HEADER
DFD:	TX
PATH:	TMSG QUEUE (RAM), TCONNECT/DISCONNECT
COMPONENTS:	PRIORITY
STRUCTURE:	TROUTE
DFD:	TX
PATH:	DETERMINE ROUTE, TCONNECT/DISCONNECT
COMPONENTS:	DESTINATION
	INTERMEDIATE NODE 1
	INTERMEDIATE NODE 2
	INTERMEDIATE NODE 3
	INTERMEDIATE NODE 4

STRUCTURE: TX C/D  
DFD: TX  
PATH: TCONNECT/DISCONNECT, INTERFACE TNC  
COMPONENTS: HEADER  
CONNECT PATH  
DISCONNECT PATH

STRUCTURE: VALIDATION VALUES  
DFD: TX  
PATH: VALID PARAMETERS, CHECK FOR COMPLETENESS  
COMPONENTS: VALID SOURCES  
VALID DESTINATIONS  
VALID DATE/TIME  
VALID AUTHORS  
VALID PRIORITIES  
VALID SECURITY CLASSIFICATIONS  
VALID MSG LENGTH

## DATA STORES

NAME: CUSTOMER'S MESSAGE (5 1/4-INCH DISK)  
DESCRIPTION: The customer's message that is stored on a 5 1/4-inch disk  
DATA FLOW IN: MSG; TRANSFER MSG  
DATA FLOW OUT: MSG; CHECK FOR COMPLETENESS  
CONTENTS: DESTINATION  
DATE/TIME  
SUBJECT  
AUTHOR  
PRIORITY  
SECURITY CLASSIFICATION  
MESSAGE TEXT

NAME: MSG STORE (RAM)  
DESCRIPTION: During generation of a message, the partial message is stored in this data store. Located in RAM. The completed message is transferred out.  
DATA FLOW IN: INPUT; DSP MSG TEMPLATE & QUERY (GENERATE)  
DATA FLOW OUT: MSG; TRANSFER MSG (GENERATE)  
CONTENTS: MSG

NAME: MSG ARCHIVE (5 1/4-INCH DISK)  
DESCRIPTION: A final storehouse for received messages. The entire message will be stored on a 5 1/4-inch disk.  
DATA FLOW IN: MSG; RCV STATISTICS (RCV)  
DATA FLOW OUT: none--by system administrator  
CONTENTS: MSG

NAME: RMSG POINTERS (RAM)  
DESCRIPTION: An ordered list, by priority, of all received messages yet to be read and/or printed out by operator. Can have up to RCVQUEUEMAX number of pointers.  
DATA FLOW IN: ORDERED R POINTER; RQUEUE (RCV)  
DATA FLOW OUT: NEXT RMSG POINTER; NOTIFY OPERATOR (RCV)  
CONTENTS: RMSG POINTER 1  
RMSG POINTER 2  
.  
RMSG POINTER RCVQUEUEMAX

NAME: RMSG QUEUE (RAM)  
DESCRIPTION: A complete file of all received messages not yet read. Stored in RAM.  
DATA FLOW IN: RMSG PKT; PARSE IN MSG (RCV)  
DATA FLOW OUT: MSG; NOTIFY OPERATOR (RCV)  
MSG; RCV STATISTICS (RCV)  
CONTENTS: MSG1  
MSG2  
MSG3

NAME: ROUTING FILE (RAM/5 1/4-INCH DISK)  
DESCRIPTION: The best current path to connect from source to destination. Also, the next best current path to connect.  
DATA FLOW IN: ROUTE; DETERMINE ROUTE (TX)  
ROUTE; UPDATE ROUTING ALGORITHM (RCV)  
DATA FLOW OUT: ROUTE; DETERMINE ROUTE (TX)  
CONTENTS: DESTINATION  
INTERMEDIATE NODE 1  
INTERMEDIATE NODE 2  
INTERMEDIATE NODE 3  
INTERMEDIATE NODE 4

NAME: TMSG QUEUE (RAM)  
DESCRIPTION: A complete file of all messages yet to be transmitted. Stored in RAM.  
DATA FLOW IN: TMSG; CHECK FOR COMPLETENESS (TX)  
DATA FLOW OUT: TMSG PRIORITY HEADER; TQUEUE (TX)  
MSG  
TCONNECT/DISCONNECT (TX)  
MSG PKT; PARSE OUT MSG (TX)  
MSG; TX STATISTICS (TX)  
CONTENTS: MSG 1  
MSG 2  
MSG 3  
  
TMSG TXQUEUEMAX

NAME: VALID PARAMETERS  
DESCRIPTION: Provides specific message elements that are acceptable.  
DATA FLOW IN: none--under system administrator control  
DATA FLOW OUT: VALIDATION VALUES; CHECK FOR COMPLETENESS (TX)  
CONTENTS: VALID SOURCES  
VALID DESTINATIONS  
VALID DATES  
VALID AUTHORS  
VALID PRIORITIES  
VALID SECURITY CLASSIFICATIONS  
VALID MESSAGE LENGTH

## Appendix E

### Test Plan

#### Table of Contents

	Page
Unit Testing . . . . .	E.2
Display Menu and Prompt Module . . . . .	E.2
Serial Port Connectivity Module . . . . .	E.3
Transmit Message Module . . . . .	E.4
Receive Message Module . . . . .	E.6
Linked List Operations Module . . . . .	E.7
Integration Tests . . . . .	E.9
Phase One - Operator Interface Cluster . . . . .	E.10
Phase Two - Communications Cluster . . . . .	E.12
Phase Three - Linked List Cluster . . . . .	E.13
Phase Four - Tie Together the Operator Interface Cluster and the Communications Cluster . . . . .	E.13
Phase Five - Tie Together the Communications Cluster and the Linked List Cluster . . . . .	E.14
Phase Six - Tie Together the Operator Interface Cluster, the Communications Cluster, and the Linked List Cluster . . . . .	E.15

## APPENDIX E

### TEST PLAN

#### Unit Testing

##### Display Menu and Prompt Module

###### Modules Functions

- display the menu listing passed in function parameter
- look for a keyboard hit and process: if selection, return; if arrow, move cursor; if valid letter, move cursor
- look for a receive request: if receive request, go to receive
- look for a transmit request: if transmit request, go to transmit

###### Testing Process

- specific area to be examined by testing
  - independent paths
  - error-handling paths
- process consists of setting up the test configuration and following the procedure

###### Testing Configuration

- IBM PC-compatible computer
- write a driver to
  - call the display menu and prompt module, passing the main menu as an initial parameter
  - embed a test cell within the display menu and prompt module that echoes the keyboard characters that are depressed on the monitor
  - process the selection returns and present the appropriate responses, i.e., when first item is selected, pass to the display menu and prompt module the submenu to display
  - process the exit request

###### Test Procedure

- invoke the driver
- depress a key
  - ensure that the key that was hit is displayed on the monitor
  - evaluate the response of the program: any key other than the up or down arrow keys, the return key, the first letter of any menu item, CTL-C should have no effect, the cursor should remain stationary

- depress the down arrow key at one-second intervals
  - the cursor should change position, moving downward to the next menu item for each key press
  - the cursor should cycle through the menu items, moving to the top menu item from the bottom menu item
- depress the up arrow key at one-second intervals
  - the cursor should change position, moving upward to the next menu item for each key press
  - the cursor should cycle through the menu items, moving to the bottom menu item from the top menu item
- depress the first letter of each menu item, pausing for one second after each key press
  - the cursor should move to the menu item associated with the letter pressed
- depress the return key
  - the position of the cursor when the return key is depressed is the control for subsequent action taken by the program
  - if the cursor was at the first menu item, a submenu should be displayed, with key press actions parroting those detailed above, from within this submenu, the return key should return the main menu independent of cursor position
  - if the cursor is at the last menu item, EXIT, the program should exit, returning control to the operating system
  - if the cursor is in any position except the first or last menu item, the monitor should clear and the main menu should be displayed
- depress CTL-C
  - the program should exit and return control to the operating system

## Serial Port Connectivity Module

### Modules Functions

- provide an ability to initialize the serial communications port
- provide an ability to send a data character out the serial communications port
- provide an ability to receive data characters automatically from the serial communications port and place in a buffer

- provide an ability to read a data character from the receive buffer
- provide an ability to close the communications port

#### Testing Process

- specific areas to be examined by testing
  - examine data structures
  - independent paths
- process consists of setting up the configuration and following the procedure

#### Test Configuration

- IBM PC-compatible computer and a TNC available
- write a driver to automatically
  - initialize the port
  - send characters from keyboard to TNC
  - receive characters from receive buffer and display on monitor
  - display all characters sent out or received from the serial port on the monitor

#### Test Procedure

- invoke driver
- send a sequence of three \*s to the TNC
- the response from the TNC should appear on the monitor and consist of the TNC sign-on message followed by the sequence command:

#### Transmit Message Module

##### Modules Functions

- query the operator for a TNC connect path
- transfer specified connect path to TNC
- look for an active connect status from TNC
- query operator for a character string to send
- transfer specified character string to TNC
- provide a disconnect request to TNC
- look for a disconnect acknowledgement from TNC

#### Testing Process

- specific area to be examined by testing
  - examine data structures
  - examine independent paths
  - examine error handling paths
- process consists of setting up the test configuration and following the procedure

#### Testing Configuration

- requires two IBM PC-compatible computers and two interconnected TNCs
- set up the destination to run the driver associated with the serial port connectivity module
- a driver to
  - call the transmit message module
  - embed test cells within the transmit message module to display the connect status response returned from the TNC, display the character string byte by byte as it is sent to the TNC, display the disconnect request being sent to the TNC, display the disconnect status response returned from the TNC

#### Testing Procedure

- invoke the driver
- at the connect request, immediately send a return
  - the response should display a message that the connect field is invalid and another connect request is made
- at the connect request, enter 30 characters
  - the response should be that the connect field is invalid and another connect request is made
- at the connect request, enter four alphanumeric characters
  - the response should be the continuation of the program
- at the request to enter a character string, enter 100 characters of alphanumeric data
  - the response should display a message that the field is invalid and another character string is made
- at the request to enter a character string, immediately send a return
  - the response should be that the program continues
- at this point, the program is run and presented with the destination call sign and a character string
  - the display shows the connect status response from the TNC; the last characters should match the destination call sign
  - once the two TNCs are connected, the display shows the character string byte by byte as it is sent out to the TNC
  - next, the disconnect request string that is sent to the TNC is displayed

- finally, the display shows the disconnect status returned from the TNC
- upon disconnect, the program returns to the operating system
- indicators of a successful data transfer can be accomplished by
  - noticing the activity of the TNC to the passing of data between TNCs
  - viewing the destination monitor for the character string received
  - requiring an exact match between the character string sent and the character string received
- run the program with a nonexistent destination call sign
  - the program should provide a display indication that the TNC is unable to connect to the destination

#### Receive Message Module

##### Modules Functions

- switch the TNC into HOST mode (see Appendix A)
- while in HOST mode, look for a receive request from the TNC
- while in HOST mode, and an active receive request condition exists, take in the character data from the receive buffer being sent by the TNC
- while in HOST mode, look for a disconnect from the TNC
- while in HOST mode, display the received message on the monitor

##### Testing Process

- specific areas to be examined by testing
  - examine data structures
  - examine independent paths
  - examine error handling paths
- process consists of setting up the test configuration and following the procedure

##### Testing Configuration

- requires two IBM PC-compatible computers and two interconnected TNCs
- set up the destination to run the driver associated with the transmit message module
- a driver to
  - call the receive message module
  - embed test cells within the receive message module to display the receive request response

from the TNC, display the character data byte by byte as it is received from the source, display the disconnect status responses from the TNC

#### Testing Procedure

- invoke the driver
- as long as the channel between the TNCs is inactive, the receive request responses will be displayed
- begin to transmit a character string from the computer that is driving the transmit message module
  - the receive request response should stop being displayed, indicating that an active receive request has been detected
  - character data should be displayed byte by byte as it is sent from the TNC
  - interleaved with the character data is the disconnect response from the TNC
  - when all the character data that was sent has arrived, the disconnect responses should no longer be displayed
  - upon a disconnect, the program returns computer control over to the operating system
- indicators of a successful data receive are
  - that all the character data sent from the source are received
  - all the character data received are in the proper sequence

#### Linked List Operations Module

##### Modules Functions

- create a doubly linked list
- display the linked list on the monitor
- provide for adding entries to the linked list
- provide for deleting entries from the linked list
- provide for sorting the entries of the linked list for a specified field
- provide for numerous fields within each entry
- provide for finding a referenced entry within the linked list and displaying on the monitor
- provide for storing the linked list on disk
- provide for loading a linked list from disk

##### Testing Process

- specific area to be examined by testing
  - examine data structures
  - examine boundary conditions
  - examine error handling paths

-process consists of setting up the test configuration and following the procedure

#### Testing Configuration

- requires an IBM PC-compatible computer
- requires a driver to
  - run the linked list operations module
  - provide a simple numbered menu of choices consisting of
    - (1) List
    - (2) Add
    - (3) Delete
    - (4) Find
    - (5) Store
    - (6) Load

#### Testing Procedure

- invoke the driver
  - note the appearance of the numbered menu
- enter the number 1 at the menu prompt
  - the display should have a message indicating the linked list is empty
- enter the number 2 at the menu prompt
  - the display should prompt through a number of field requests for entries and check that the number of entries is consistent with the number specified in the program
- enter the number 1 at the menu prompt
  - the display should show the linked list entry that was just completed
- enter the number 2 at the menu prompt
  - the display should prompt through a number of field requests for entries and check that the number of entries is consistent with the number specified in the program
- enter the number 2 at the menu prompt
  - the display should show the two linked list entries just completed and ensure that the listing is sorted according to the order dictated by the program
- enter the number 3 at the menu prompt
  - the display should show the first field of each linked list entry in the properly sorted order
  - a prompt should request the name of the first field of the entry to delete
- enter the first name listed
- enter the number 1 at the menu prompt
  - the linked listing should not contain the entry that was just deleted

- enter the number 2 at the menu prompt and enter another entry to the linked list
- enter the number 4 at the menu prompt
  - a prompt should appear requesting the first field of the entry to find
- enter the first field of the recently completed entry
  - the entire listing of the requested entry should be displayed for five seconds, followed by the display of the main menu
- enter the number 5 at the menu prompt
  - the light on the disk drive should light, indicating that the program is storing the linked list on disk in the file specified by the program
- enter the number 3 at the menu prompt repeatedly until all linked list entries have been deleted
- enter the number 6 at the main menu prompt
  - the light on the disk drive should light, indicating that the program is reading from the linked list disk file specified by the program
- enter the number 1 at the main menu prompt
  - the linked listing should display the entries that were stored on disk

#### Integration Tests

- broken into six phases based on function and complexity
- performance objectives form the basis for the test measures
- test measures of performance will be based on equivalence testing

#### Phase One

##### Operator Interface Cluster

##### Performance Objectives

- tailored menu listings to system requirements
- when cluster is built, it forms a menu-driven operator interface that can accommodate all functions dictated by requirements
- the actual functions called by a menu selection will be stubs that display an indication that the stub was entered
- stubs return control to the correct level of menu that would occur in operation when stubs are replaced by the operational function

#### Principal Module Components

- display menu and prompt module

#### Modules Formed

- operator interface state module

#### Testing Process

- testing process consists of setting up the test configuration and following the procedure
- specific areas to be examined by testing
  - valid and invalid input
  - only menu listings called for in the design are incorporated
  - a specific menu selection leads to the correct program stub that displays the correct progress indicator

#### Testing Configuration

- requires an IBM PC-compatible computer
- requires a driver only to call the operator interface module

#### Test Procedure

- invoke the driver
- check that the display shows a menu header and a main menu listing
  - check that the main menu listing conforms to design (see Chapter III)
- check that the function keys specified during unit testing of the display menu and prompt module are operating properly
- choose the first main menu item
- check that the submenu displayed or the stub indicator displayed conforms to design (see Chapter III)
- choose the return to main menu choice from the submenu
- repeat the last three steps for all the remaining menu items
- repeat the last three steps for all submenu items of all main menu selections
- repeat the last three steps for all subsubmenu items of all submenu selections

#### Phase Two

##### Communications Cluster

#### Performance Objectives

- provide for automatic transmission and receiving of an operator-specified character string to an operator-specified destination

- query the operator for the station call sign
- query the operator for a transmit request
- when the operator selects a transmit request, prompt for a destination and a character string
- if no transmit request is made, the program monitors the TNC for a connect and displays any received characters on the display

#### Principal Module Components

- receive message module
- transmit message module

#### Modules Formed

- state transition manager module
- receive state module
- transmit state module

#### Testing Process

- testing process consists of setting up the test configuration and following the test procedure
- specific areas to be examined by testing
  - ability of the program to switch from the receive mode to the transmit mode
  - ability of the program to switch from the transmit mode to the receive mode

#### Test Configuration

- requires two IBM PC-compatible computers and two interconnected TNCs
- requires a driver to
  - call the state transition manager module

#### Test Procedure

- invoke the driver on both the computers
- check that both computers display requests for the operator to signal a transmit request
- enter a transmit request at one of the computers and process the transmit
- check that the other computer received the transmitted character string
- check that the character strings that were transmitted and received are composed of the same characters and sequence of those characters
- check to see whether the transmitting computer returns to the receive ready mode by looking for the transmit request on the display
- check to see that the receiving computer returns from the receiving mode to the transmit request mode by looking for the transmit request on the display
- enter a transmit request from the computer that just received a character string

- check that the character string is transmitted correctly to the other computer
- check to see whether the transmitting computer returns to the receive ready mode by looking for the transmit request on the display
- check to see that the receiving computer returns from the receiving mode to the transmit request mode by looking for the transmit request on the display

### Phase Three

#### Linked List Cluster

##### Performance Objectives

- multiple linked list tailored to requirements
- transfer an entry from one linked list to another
- tailor the linked list entries to requirements
- display one of the multiple linked lists
- provide for a display of one entry of a chosen linked list

##### Principal Module Components

- linked list operations module

##### Modules Formed

- initialize linked lists module
- access archive module
- generate message module
- update routing tables
- update message archive
- store linked lists

##### Testing Process

- testing process consists of setting up the test configuration and following the test procedure
- specific areas to be examined by testing
  - all linked lists formed
  - ability to work with any linked lists

##### Test Configuration

- requires an IBM PC-compatible computer
- requires a driver to
  - initialize five linked lists
    - transmit queue linked list
    - transmitted linked list
    - received linked list
    - generate message linked list
    - routing table linked list
  - provide a limited menu listing to include

- Operate on linked list
- List
- Delete
- Add
- Store
- Load
- move generate message linked list entries into the transmit queue to the transmitted linked list, copying into the received linked list and using the generate message entry field to update the routing table linked list

#### Test Procedure

- invoke the driver
- from the menu, choose each linked list, in turn, and verify that all listings are empty
- add an entry to the generate message linked list
- verify that all linked lists now contain the generate message entry except for the routing table, which contains the correct source and destination fields from the generate message entry

#### Phase Four

Tie Together the Operator Interface Cluster and the Communications Cluster

#### Performance Objectives

- from a menu selection, enable automatic connection and transfer of a character string
- destination needs to automatically receive character string while in any menu configuration

#### Principal Module Components

- state transition manager module
- operator interface state module
- receive state module
- transmit state module

#### Modules Formed

- none

#### Testing Process

- testing process consists of setting up the test configuration and following the test procedure
- specific area to be examined by testing
- ability of the modules to work together

#### Testing Configuration

- requires two IBM PC-compatible computers and two interconnected TNCs

- requires a driver to
  - call the operator interface state module

#### Testing Procedure

- invoke the driver at each of the computers
- at one of the computers, choose the transmit option from among the menu listings
- enter the requested information to establish a transmission
- check that the other computer receives the message from the monitor display
- check that after transmission, the program returns to the main menu
- check that the other computer returns to the main menu after receiving a message
- switch the process to transmit and receive on opposite computers
- check that the message is transmitted properly to the receiver's monitor
- check that the program returns to the main menu after transmission and receipt of the character string

#### Phase Five

#### Tie Together the Communications Cluster and the Linked List Cluster

#### Performance Objectives

- a message structure linked list entry replaces the character string
- the destination is taken from the linked list entry
- the destination points to a routing table linked list entry for the connect path
- the destination updates the appropriate routing table linked list entry upon receipt of a message structure linked list entry
- the destination stores the received message structure linked list entry in the received message queue linked list

#### Principal Module Components

- state transition manager module
- receive state module
- transmit state module
- initialize linked lists module
- generate message module
- update routing tables
- update message archive

#### Modules Formed

- receive state module
- transmit state module

#### Testing Process

- testing process consists of setting up the test configuration and following the test procedure
- specific area to be examined by testing
  - interaction between the linked list modules and the transmit and receive modules

#### Test Configuration

- requires two IBM PC-compatible computers and two interconnected TNCs
- requires a driver to
  - call the state transition manager module
  - provide an abbreviated menu

#### Test Procedure

- invoke the driver at both computers
- select the menu item to view all linked lists and ensure they are empty
- select the menu item to generate a message
- generate a message
- select the menu item to transmit a generated message
- check that the message is in the process of being transmitted by monitoring the TNC indicator lamps
- after transmission of the message, select the menu option to see that the transmitted message updated the transmitted linked list at the source computer
- after transmission of the message, select the menu option to see that the transmit queue linked list is clear
- after receiving the message, choose the menu item to see that the recently received message has updated the received message linked list
- reverse the process to the other computers and go through the test procedure again

#### Phase Six

Tie Together the Operator Interface Cluster, the Communications Cluster, and the Linked List Cluster

#### Performance Objectives

- perform all the functions detailed by design (see Chapter III)

#### Principal Module Components

- all

#### Modules Formed

- none

#### Testing Process

- testing process consists of setting up the test configuration and following the test procedure
- specific area to be examined by testing
- interaction correct between all called modules

#### Test Configuration

- requires two IBM PC-compatible computers and two interconnected TNCs
- no driver is required, since the program is now in final form
- no stubs are used
- there is still an embedded code that displays program progress indicators on the monitor

#### Test Procedure

- run the program at each computer
- check that full capabilities to transmit and receive a module exist
- check that the linked lists are properly updated after each transmission

## Appendix F

### Test Results

#### Table of Contents

	Page
Verification Tests . . . . .	F.2
Unit Tests . . . . .	F.2
Display Menu and Prompt Module . . . . .	F.2
Serial Port Connectivity Module . . . . .	F.3
Transmit Message Module . . . . .	F.3
Receive Message Module . . . . .	F.4
Linked List Operations Module . . . . .	F.5
Integration Tests . . . . .	F.6
Phase One - Operator Interface Cluster. . . . .	F.6
Phase Two - Communications Cluster . . . . .	F.7
Phase Three - Linked List Cluster . . . . .	F.8
Phase Four - Tie Together the Operator Interface Cluster and the Communications Cluster . . . . .	F.9
Phase Five - Tie Together the Communications Cluster and the Linked List Cluster . . . . .	F.9
Phase Six - Tie Together the Operator Interface Cluster, the Communications Cluster, and the Linked List Cluster . . . . .	F.10
Validation Tests . . . . .	F.11
Alpha Testing . . . . .	F.11

Appendix F  
TEST RESULTS

Verification Tests

Unit Tests

Display Menu and Prompt Module

Test Procedure Results

- invoked the driver
  - menu listing was displayed
- depressed a "q"
  - a "q" was displayed
  - no apparent effect on program
- depressed the down arrow key at one-second intervals
  - the cursor changed position, moving downward to the next menu item for each key press
  - the cursor cycled through the menu items, moving to the top menu item from the bottom menu item
- depressed the up arrow key at one-second intervals
  - the cursor changed position, moving upward to the next menu item for each key press
  - the cursor cycled through the menu items, moving to the bottom menu item from top menu item
- depressed the first letter of each menu item, pausing for one second after each key press
  - the cursor moved to the menu item associated with the letter pressed
- depressed the return key while the cursor was at the first menu item
  - a submenu was displayed
  - repeating the key press actions detailed above resulted in the correct responses
  - upon selection of the RETURN TO MAIN MENU option, the program displayed the main menu
- depressed the return key while the cursor was at the second menu item listed
  - the monitor cleared and the main menu reappeared

- depressed the return key while the cursor was at the last menu item, EXIT
  - the program exited, returning control to the operating system
- invoked the driver
- depressed CTL-C
  - the program exited and returned control to the operating system

#### Serial Port Connectivity Module

##### Test Procedure Results

- invoked the driver
- entered a sequence of three \*s from the keyboard
  - the monitor displayed the TNC sign-on message followed by the sequence "cmd:"

#### Transmit Message Module

##### Test Procedure Results

- invoked the driver
  - queried the operator for a connect path
- depressed the return key
  - the monitor displayed a message that the connect field was invalid
  - the operator was queried for another connect path
- depressed a random sequence of 30 characters
  - the monitor displayed a message that the connect field was invalid
  - the operator was queried for another connect path
- depressed the character sequence N3NN, the call sign of the other TNC
  - the monitor displayed a request for the character string to transmit
- depressed a random character string of 101 characters
  - the monitor displayed a message indicating that the character string was invalid
  - the monitor displayed a query to the operator for another character string

- depressed the return key
  - the program began to process the transmit request
  - the monitor displayed connect status responses from the TNC
  - the last characters were N8NN
  - connect status responses stopped being displayed
  - the character string was displayed on the monitor
  - the disconnect request sent to the TNC was displayed
  - the monitor displayed the disconnect status responses from the TNC
  - the disconnect status response stopped being displayed
  - the operating system prompt appeared
- invoked the driver
- provided the input data to send a character string of 25 random characters to N8NN
- monitored the call progress indicators on the monitor
- compared the random character string transmitted to the character string received
  - the character strings matched exactly

#### Receive Message Module

##### Test Procedure Results

- invoked the driver
  - the TNC responses to the receive requests initiated by the program were displayed
- transmitted a character string from a computer to the computer running the receive message module
  - the receive request responses stopped being displayed
  - character data were displayed byte by byte as they were sent from the TNC
  - the disconnect response from the TNC was interleaved with the character data
  - the character data and interleaved disconnect response from the TNC stopped being displayed
  - the character strings transmitted and received matched exactly
  - the monitor displayed the operating system prompt

## Linked List Operations Module

### Test Procedure Results

- invoked the driver
  - a numbered menu appeared
- entered the number 1 at the menu prompt
  - the display had a message indicating that the linked list is empty
- entered the number 2 at the menu prompt
  - the display prompted for an entry to the first field
- an iterative process began that first entered an excessively large character string, then upon getting a field error indication, a proper character string was entered causing the program to advance to the next field
- entered the number 1 at the menu prompt
  - the display showed the linked list that was just entered
- entered the number 2 at the menu prompt
  - the display prompted for an entry to the first field
- an iterative process began that first entered an excessively large character string, then upon getting a field error indication, a proper character string was entered causing the program to advance to the next field
- entered the number 1 at the menu prompt
  - the monitor display showed the two linked list entries just completed
  - the listings are sorted according to the order dictated by the program
- entered the number 3 at the menu prompt
  - the monitor displayed the first field of each linked list entry in the properly sorted order
  - a prompt requested the name of the first field of the entry to delete
- entered the first name listed
- entered the number 1 at the menu prompt
  - the linked listing no longer contained the entry that was just deleted
- entered the number 2 at the menu prompt, entering another entry to the linked list
- entered the number 4 at the menu prompt
  - a prompt appeared requesting the first field of the entry to find

- entered the first field of the recently completed entry
  - the entire listing of the requested entry was displayed for five seconds, followed by the display of the numbered menu
- entered the number 5 at the menu prompt
  - the disk drive light illuminated
- entered the number 3 at the menu prompt repeatedly until all linked list entries were deleted
- entered the number 6 at the main menu prompt
  - the disk drive light illuminated
- entered the number 1 at the main menu prompt
  - the monitor displayed the linked listing entries that were stored on disk
  - the linked list matched the previous linked list that was deleted

## Integration Tests

### Phase One - Operator Interface Cluster

#### Test Procedure Results

- invoked the driver
  - the monitor displayed a menu header and a main menu listing
  - the main menu listing conformed with design
  - the functional keys, as specified during unit testing of the display menu and prompt module operated properly
- selected the first main menu item, TRANSMIT MESSAGE
  - the transmit menu was displayed
  - the transmit menu displayed conformed with design
- from the transmit menu listing, selected all items in turn
  - each selection produced the proper response via display of another menu or a stub
- from the transmit menu listing, the RETURN TO MAIN MENU item was chosen
  - the program returned to the main menu

- selected the second main menu item, GENERATE A MESSAGE
  - the monitor displayed the information header for generating a message
  - the monitor prompted through message fields for input
  - the main menu was displayed after all message fields had been entered
- selected the third main menu item, ACCESS ARCHIVED MESSAGE FILE
  - the archive menu was displayed
  - the archive menu conformed to design
- from the archive menu listing, selected all items in turn
  - each selection produced the proper response via display of another menu or a stub
  - the submenu and routing table submenu performed as specified
- from the archive menu, the RETURN TO MAIN MENU item was chosen
  - the program returned to the main menu
- selected the fourth menu item, EXIT PROGRAM
  - the disk drive light illuminated briefly
  - the operating system prompt appeared

## Phase Two - Communications Cluster

### Test Procedure Results

- invoked the driver on both computers
  - both computers displayed requests for the operator to signal a transmit request
- entered a transmit request at one of the computers and processed the transmit
  - the other computer received the transmitted character string
  - the character strings which were transmitted and received were composed of the same characters and sequence
  - the transmitting computer returned to the transmit request mode
  - the receiving computer returned from the receiving mode to the transmit request mode

- entered a transmit request from the computer that was the recent receiving computer
  - the other computer received the transmitted character string
  - the character strings which were transmitted and received were composed of the same characters and sequence
  - the transmitting computer returned to the transmit request mode
  - the receiving computer returned from the receiving mode to the transmit request mode

### Phase Three - Linked List Cluster

#### Test Procedure Results

- invoked the driver
  - the menu was displayed
- the list menu item was selected for all linked lists
  - all linked lists were empty
- the add menu item was selected
  - the program displayed prompts for the operator to enter generate message fields
- the fields were entered for the generate message entry
  - after the generate message function entries were completed, the program displayed the main menu
- the list menu item is selected for all linked lists
  - all linked lists except the transmit queue had the recently entered message
- the store menu item is selected
  - the disk drive light illuminated for a brief period
- the delete menu item was selected recursively to clear all linked lists
- the load menu item was selected
  - the disk drive light illuminated for a brief period
- the list menu item was selected recursively for all linked lists
  - all linked lists had the message that was previously entered

#### Phase Four - Tie Together the Operator Interface Cluster and the Communications Cluster

##### Test Procedure Results

- invoked the driver at each of the computers
- at one of the computers, the transmit option was chosen, and a connect path and character were entered
  - the transmitting computer began to display connect status responses from the TNC
  - the receiving computer was displaying connect status responses from the TNC, but stopped and began to display disconnect
  - the character string was displayed byte by byte at the transmitting computer
  - at the receiving computer, the displaying of the received character string is interleaved with the disconnect responses byte by byte
  - at the transmitting computer, all the character string characters had been displayed
  - at the receiving computer, all the character string characters were displayed in proper sequence
  - the transmitting computer began to display disconnect responses from the TNC
  - the receiving computer continued to display disconnect responses from the TNC
  - the transmitting and receiving computer displayed a final disconnect response different than the previous responses
  - both computers returned to the main menu
- the process was then reversed; the computer that recently transmitted became the receiver, and the computer that was the receiver became the transmitter
  - results were the same as presented above

#### Phase Five - Tie Together the Communications Cluster and the Linked List Cluster

##### Test Procedure Results

- invoked the driver at both computers
  - an abbreviated menu was displayed on each computer monitor

- selected the menu item to view linked lists recursively
  - all linked lists empty
- selected the menu item to generate a message
- generated a message
- selected the menu item to transmit a generated message
  - TNC indicator lamps indicated that the message was being transmitted
  - a message appeared on both computer monitors to indicate that a message was being received or transmitted
  - the message indicating that a message was being received or transmitted no longer appeared on the computer monitors
- at the transmitting computer, the list menu item was selected and the transmitted linked list was chosen
  - the transmitted linked list contained the message that was recently transmitted
- at the receiving computer, the list menu item was selected and the received linked list was chosen
  - the received linked list contained the message that was recently received
- the transmitting and receiving computers were reversed and the test procedure was rerun
  - exactly the same results as above were obtained

Phase Six - Tie Together the Operator Interface Cluster, the Communications Cluster, and the Linked List Cluster

#### Test Procedure Results

- ran the program at each computer
  - the main menu appeared on each computer monitor
- selected the access archived message file menu item at each computer and checked each linked list
  - all linked lists were empty
- selected the generate message menu item at each computer and generated a message
  - the disk drive light illuminated for a brief period

- selected the transmit message menu item at one of the computers
  - the disk drive light illuminated for a brief period
  - a message was displayed indicating that the transmit queue was not empty, but the transmit flag was disabled
- from within the transmit menu, selected the begin transmit menu item
  - a message was displayed on both computers indicating that a transmission or receiving of a message was occurring
  - the message indication was erased
  - the main menu was displayed
- from the main menu of both computers, the access archived message file was selected
- due to the recent message transmission, the linked lists at the transmitting and receiving computer were checked for update
  - the linked lists were updated
- the transmit message process was redone, switching the transmit and receive computers
  - the results were exactly as obtained above
- at both computers, the first routing path of the routing table linked list was purposely altered with random characters to check whether the routing table was being updated on transmit and receive
- at one computer, a transmit was processed
  - the routing table was correctly updated at both the transmit and receive computers

## Validation Tests

### Alpha Testing

Table F-I. Alpha Phase Validation Test, Expected Software Responses, Actual Responses

<u>Req</u>	<u>Validation Test</u>	<u>Expected Software Response</u>	<u>Actual Response</u>
(1)	Send a message to all nodes.	Message would be received correctly at all nodes.	Message was received correctly at all nodes.
(2)	Place all message information into message.	Complete message would transfer to a 5 1/4-inch disk.	Complete message was transferred to a 5 1/4-inch disk.
(3)	Compose a message of more than one and a half pages of text.	Only one and a half pages of text would transfer to disk.	Operator was informed upon entry that the maximum message length was exceeded and new text was requested.
(4)	Send a message to known destination that has no direct connection.	Message would arrive at destination via an alternate path.	Continuing to be tested.
(5)	Send a message.	Message would arrive complete.	Message arrived complete.
(6)	Continuously enter messages into the transmit queue with a single destination until a queue full indication.	The number of messages entered would be equal to, or slightly greater than, TXQUEUEMAX, due to transmit during entry.	The queue full indicator was the out-of-memory indicator. The number of messages that can be entered into the transmit queue was limited by the computer's memory.

Table F-I (Cont'd)

<u>Reg</u>	<u>Validation Test</u>	<u>Expected Software Response</u>	<u>Actual Response</u>
(7)	Send a message to a known off-line node.	Indicator would inform the operator that the message did not get through in TIMEOUT minutes or TRYOUT tries.	Indicators informed the operator that the message did not get through in TIMEOUT minutes.
(8)	Fill the transmit queue with a low-priority message, then immediately enter a higher priority message.	The higher priority message would be sent before the lower priority message.	The higher priority message was sent before the lower priority message.
(9)	Send a message to a node having an unattended operator position.	Message would be received at the destination and placed in the received archive file.	Message was received at the destination and placed in the received message linked listing.
(10)	One node is processing the entry of a message. A second node is sending the first node a message.	Messages transmitted to the first node would be received.	Messages transmitted to the first node were received. The entry process was fast enough to finish and receive.
(11)	One node continuously processes message entries, with a second node as the destination. The second node is able to receive messages.	The second node would receive the messages.	The second node received the messages.

Table F-I (Cont'd)

<u>Req</u>	<u>Validation Test</u>	<u>Expected Software Response</u>	<u>Actual Response</u>
(12)	Activate operator interface. Repeatedly call up the help option for each operator option.	The operator interface would be menu-driven. The help feature would detail all major operator options.	The operator interface was menu-driven. The help option was not available.
(13)	Input a semi-completed message header, e.g., leave out the destination, source, author, etc.	Message would be detected as incomplete and cause an incomplete indication at the operator position.	The message completeness option was not available.
(14)	Continuously enter message into the transmit queue with a single destination until a queue full indication. Count the number of messages entered.	All messages entered would be received. Message receipt by the operator would proceed through the received message queue.	All messages entered were received. Messages received were automatically put in the received message queue.
(15)	Activate the generate program. Enter a message.	The screen would display a message template and a cursor query for input. The message would be stored on a 5 1/4-inch disk.	The screen displayed a crude message template and a cursor query for input. The message was stored on a 5 1/4-inch disk.

Table F-I (Cont'd)

<u>Req</u>	<u>Validation Test</u>	<u>Expected Software Response</u>	<u>Actual Response</u>
(16)	Activate the generate message program. Repeatedly call up the help feature for all customer options, message fields.	The generate message program would be menu-driven with accurate, clearly worded, detailed help features.	The generate message program was menu-driven. Help features were not implemented.
(17)	Activate the generate message program.	The generate message program would have destination, author, date, time, priority, security, and message text fields.	The generate message program had destination, author, date, time, priority, security, and message text fields.
(23)	Activate the generate message program on an IBM PC computer that runs MS-DOS 3.X.	The generate message program would run correctly.	The generate message program ran correctly.

## Appendix G

### Table of Contents

State Transition Manager Module  
Initialize System State Module  
Initialize Port Module  
Initialize TNC Module  
Initialize Linked List Module  
Operator Interface State Module  
Display Menu and Prompt Module  
Prepare a Message for Transmit Module  
Access Archive Module  
Generate Message Module  
Receive State Module  
Receive Message Module  
Update Routing Table Module (Receive)  
Update Message Archive Module (Receive)  
Transmit State Module  
Transfer Message Module  
Update Routing Table Module (Transmit)  
Update Message Archive (Transmit)  
Reset System State Module  
include/netos.h file

The code associated with each module is the property of the United States Air Force. A listing of the code can be made available to authorized personnel by contacting Captain William Taris, whose permanent mailing address is listed in the Vita.

VITA

Captain William J. Taris [REDACTED]  
[REDACTED]  
[REDACTED]

[REDACTED] in 1972 [REDACTED] was awarded a full tuition scholarship to attend Control Data Institute's 10-month technical school. Following the completion of this training in May 1973, he enlisted in the U.S. Army and served with the 440th Signal Battalion until September 1976. After completing an Associate Degree at Moraine Valley Community College, he was employed by the Wescom Telecommunications Company until attending Illinois State University. Upon graduation from Illinois State University, from which he received the degree of Bachelor of Science in Chemistry, he was employed by the Illinois Water Treatment Company until May 1982. He was awarded a U.S. Air Force commission upon graduation from Officer Training School on 26 August 1982. His first Air Force assignment was to attend Louisiana Tech University to pursue the degree of Bachelor of Science in Electrical Engineering. Following the successful completion of this assignment, in June 1984, he was assigned to the 1842 Electronics Engineering Group, Air Force Communications Command, Scott AFB, Illinois, until March 1987. While serving his assignment at Scott AFB, he earned the degree of Master of Arts in Procurement and Materials Management from Webster University. He then attended Squadron Officer School en route to his Air Force Institute of Technology assignment to pursue the degree of Master of Science in Electrical Engineering.

[REDACTED]

# REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/EE/88D-53			7a. NAME OF MONITORING ORGANIZATION		
6a. NAME OF PERFORMING ORGANIZATION School of Engineering		6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7b. ADDRESS (City, State, and ZIP Code)		
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433			9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Logistics Cmd.		8b. OFFICE SYMBOL (If applicable) AFLC/SCMX	10. SOURCE OF FUNDING NUMBERS		
8c. ADDRESS (City, State, and ZIP Code) AFLC/SCMX Wright-Patterson AFB, Ohio 45433			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
11. TITLE (Include Security Classification) See Box 19			WORK UNIT ACCESSION NO.		
12. PERSONAL AUTHOR(S) William J. Paris, B.S.E.E., Capt, USAF					
13a. TYPE OF REPORT MS THESIS		13b. TIME COVERED FROM _____ TO _____		14. DATE OF REPORT (Year, Month, Day) 1988 December	
15. PAGE COUNT 179					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Packet Radio, Communications Networks		
25	02				
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
<p>Title: Design and Development of a Computer-Based Message Transfer System for the Air Force Logistics Command Packet Radio Network</p> <p>Thesis Chairman: LTC Albert B. Garcia</p> <p>Abstract on next page</p> <p style="text-align: right;"><i>Supervised 12 Jan 1989</i></p>					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL LTC Albert B. Garcia			22b. TELEPHONE (Include Area Code) 513-255-3576		22c. OFFICE SYMBOL AFIT/ENG

UNCLASSIFIED

Item 19 (Cont'd.)

Abstract The Air Force Logistics Command (AFLC) Packet Radio Network (PRN) is a specialized communications network that enables communication between eight logistics command centers throughout the continental United States. The PRN communicates by transferring a message from a microcomputer onto a broadcast radio channel. This thesis effort designs, and partially develops the design for, a computer-based message transfer system that operates the PRN. First, system requirements are established, a logical system model is constructed, and validation tests are detailed. The computer-based message transfer system has two fundamental requirements--that it be easy to use and that it provide automatic routing through the network. Next, the design is built supporting a hierarchical program structure, modularity and information hiding. After the computer code, detailed by design, is written, it is tested. Testing involves a comparison of the validation tests detailed earlier with the computer program's operation. The results show that this thesis effort resulted in an operational computer-based message system for the PRN that satisfies the two fundamental requirements of ease of use and automatic routing.

UNCLASSIFIED